IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Appl. No.	:	09/759728
Applicant	:	Elliot Schwartz
Filed	:	01/11/2001
TC/A.U.	:	2131
Examiner	:	MOORTHY, ARAVIND K
Confirmation No.	:	2453
Docket No.	:	005168.P001
Customer No.	:	40418

Title : METHOD AND APPARATUS FOR FIREWALL TRAVERSAL

COMMUNICATED VIA EFS on 15 August 2007

APPELLANT'S APPEAL BRIEF Under 37 C.F.R. § 41.37

Dear Sir:

Applicant (Appellant) hereby submits this Appeal Brief pursuant to 37 C.F.R. § 41.37 in connection with the above-referenced application and respectfully requests consideration by the Board of Patent Appeals and Interferences for allowance from a fourth rejection decision by the Examiner. The Examiner's fourth rejection decision ("Office Action" or "Office") was mailed on August 10, 2006 and rejected all claims (1-26). Applicant also submits herewith a credit card authorization or payment in the amount of $250 as the fee for filing an Appeal Brief required by 37 C.F.R. § 41.20(b)(2). Applicant submitted a Notice of Appeal that was received on February 15, 2007.

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest of Appellant is Digi International, Incorporated.

II. RELATED APPEALS AND INTERFERENCES

Appellant is unaware of any related appeals or interferences.

1) There are no prior or pending interferences.

2) There are no pending appeals before the USPTO.

3) There is no judicial proceeding related to Appellant's instant application or related applications.

III. STATUS OF THE CLAIMS

Claims 1-26 are pending in the application. No claims have been allowed.

All claims are on appeal.

All claims have been rejected by the Examiner as follows.

a) Claims 1-16 and 22-26 are rejected under 35 U.S.C. 112, first paragraph.

b) Claims 1-4, 10, 12, 14-16 and 22-26 are rejected under 35 U.S.C. 102(b) as being anticipated by Vellanki et al U.S. Patent No. 5,999,979.

c) Claim 4 is rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 1 above, and further in view of Bhide et al U.S. Patent No. 5,852,717.

d) Claims 5-9 are rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 1 above, and further in view of Fuh et al U.S. Patent No. 6,609,154 B1.

e) Claims 11 and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 1 above, and further in view of Fuh et al U.S. Patent No. 6,609,154 B1.

f) Claim 16 is rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 14 above, and further in view of Linden et al U.S. Patent No. 6,549,773 B1.

g) Claims 24 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 22 above, and further in view of Fuh et al U.S. Patent No. 6,609,154 B1.

h) Claim 26 is rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 22 above, and further in view of Montenegro U.S. Patent No. 6,233,688 B1.

IV. STATUS OF AMENDMENTS

No amendments have been filed after receipt of the fourth rejection.

//

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

Appellant has indicated below representative Figures/Specifications – not all are included so as to keep the claims brief. **Appellant has indicated in larger font the four independent claims (1, 14, 17, and 22)**.

Appellant will first address the summary of the 4 independent claims (1, 14, 17, and 22) followed by an abbreviated summary of the dependent claims.

Independent claim 1:

1. A method for traversing a firewall, comprising:

initiating a first connection to go through said firewall (Figure 4 at 402. Specification: page 13, lines 3-4.; Figure 5 at 506, 508 first time through. Specification: page 14, line 18 to page 15, line 9.)**;**

evaluating the first connection for a response from a remote system indicating a successful first connection (Figure 4 at 404 Yes arrow. Specification page 13, lines 20-21.; Figure 5 at 510 Yes arrow. Specification: page 14, line 18 to page 15, line 9.)**;**

initiating a second connection (Figure 4, 408. Specification page 13, lines 21-22.; Figure 5 at 506, 508 second time through. Specification: page 14, line 18 to page 15, line 9.) **to go through said firewall if a successful first connection is not**

established (Figure 4, 404 No arrow. Specification page 13, lines 21-22.; Figure 5 at 510 No arrow. Specification: page 14, line 18 to page 15, line 9.)**, wherein said second connection is different than said first connection** (Figure 4 at 408 HTTP. Specification page 13, lines 21-22. v. Figure 4 at 402 TCP port X. Specification: page 13, lines 3-4.)**, ;**

evaluating the second connection for a response from a remote system indicating a successful second connection (Figure 4, 410 Yes arrow. Specification page 12, line 14 to page 14 line 17)**;**

initiating a third connection (Figure 4, 412. Specification page 12, line 14 to page 14 line 17.) **to go through said firewall if a successful second connection is not established** (Figure 4, 410 No arrow. Specification page 12, line 14 to page 14 line 17)**, wherein said third connection is different than said second connection and said first connection** (Figure 4, 412 HTTP proxy. Specification page 12, line 14 to page 14 line 17. v. Figure 4 at 408 HTTP. Specification page 13, lines 21-22. v. Figure 4 at 402 TCP port X. Specification: page 13, lines 3-4.)**; and**

evaluating the third connection for a response from a remote system indicating a successful third connection (Figure 4, 414 Yes arrow. Specification page 12, line 14 to page 14 line 17)**.**

//

Independent claim 14:

14. A machine-readable medium (Figure 2 at 206, 208, 210. Specification: page 7 lines 11-16, page 8 lines 5-7; Figure 6 at 603. Specification page 18 lines 3-11) **having stored thereon instructions** (Specification: page 7 lines 11-16; page 24, line 11.)**, which when executed by a processor** (Figure 2 at 204. Specification: page 8, line 6.)**, causes said processor to perform the following:**

initiate a first connection to go through a firewall (Figure 4 at 402. Specification: page 13, lines 3-4.; Figure 5 at 506, 508 first time through. Specification: page 14, line 18 to page 15, line 9.)**;**

evaluate the first connection for a response from a remote system indicating a successful first connection (Figure 4 at 404 Yes arrow. Specification page 13, lines 20-21.; Figure 5 at 510 Yes arrow. Specification: page 14, line 18 to page 15, line 9.)**;**

initiate a second connection (Figure 4, 408. Specification page 13, lines 21-22.; Figure 5 at 506, 508 second time through. Specification: page 14, line 18 to page 15, line 9.) **to go through said firewall if a successful first connection is not established** (Figure 4, 404 No arrow. Specification page 13, lines 21-22.; Figure 5 at 510 No arrow. Specification: page 14, line 18 to page 15, line 9.)**, wherein said second connection is different than said first connection** (Figure 4 at 408 HTTP.

Specification page 13, lines 21-22. v. Figure 4 at 402 TCP port X. Specification: page 13, lines 3-4.)**;**

evaluate the second connection for a response from a remote system indicating a successful second connection (Figure 4, 410 Yes arrow. Specification page 12, line 14 to page 14 line 17)**;**

initiate a third connection (Figure 4, 412. Specification page 12, line 14 to page 14 line 17.) **to go through said firewall if a successful second connection is not established** (Figure 4, 410 No arrow. Specification page 12, line 14 to page 14 line 17)**, wherein said third connection is different than said second connection and said first connection** (Figure 4, 412 HTTP proxy. Specification page 12, line 14 to page 14 line 17. v. Figure 4 at 408 HTTP. Specification page 13, lines 21-22. v. Figure 4 at 402 TCP port X. Specification: page 13, lines 3-4.)**; and**

evaluate the third connection for a response from a remote system indicating a successful third connection (Figure 4, 414 Yes arrow. Specification page 12, line 14 to page 14 line 17)**.**

//

Independent claim 17:

17. A firewall traversal system comprising:

a main system (Figure 6 at 602. Specification page 18, lines 3-11.) **coupled to storage** (Figure 6 at 604. Specification page 18, lines 3-11.)**;**

a communication subsystem (Figure 6 at 606. Specification page 18, lines 3-11.) **coupled to the main system** (Figure 6 at 602. Specification page 18, lines 3-11.) **and a communication medium** (Figure 6 at 612. Specification page 18, lines 3-11.) **on one side of a firewall** (Figure 3 at 310. Specification page 10, line 21.)**;**

a packet examining subsystem (Figure 6 at 608. Specification page 18, lines 3-11.) **coupled to the communication subsystem** (Figure 6 at 606. Specification page 18, lines 3-11.)**; and**

a database system (Figure 6 at 610. Specification page 18, lines 3-11.) **coupled to the packet examining subsystem** (Figure 6 at 608. Specification page 18, lines 3-11.) **and the main system** (Figure 6 at 602. Specification page 18, lines 3-11.)**.**

//

Independent claim 22:

22. A method for traversing a firewall, comprising:

means for initiating a first connection to go through said firewall (Figure 2. Specification page 8 line 3 to page 9 line 17.; Figure 3. Specification page 10 line 15 to page 11 line 23.; Figure 4 at 402. Specification: page 13, lines 3-4.; Figure 5 at 506, 508 first time through. Specification: page 14, line 18 to page 15, line 9.)**;**

means for evaluating the first connection for a response from a remote system indicating a successful first connection (Figure 2. Specification page 8 line 3 to page 9 line 17.; Figure 3. Specification page 10 line 15 to page 11 line 23.; Figure 4 at 404 Yes arrow. Specification page 13, lines 20-21.; Figure 5 at 510 Yes arrow. Specification: page 14, line 18 to page 15, line 9.)**;**

means for initiating a second connection (Figure 2. Specification page 8 line 3 to page 9 line 17.; Figure 3. Specification page 10 line 15 to page 11 line 23.; Figure 4, 408. Specification page 13, lines 21-22.; Figure 5 at 506, 508 second time through. Specification: page 14, line 18 to page 15, line 9.) **to go through said firewall if a successful first connection is not established** (Figure 2. Specification page 8 line 3 to page 9 line 17.; Figure 3. Specification page 10 line 15 to page 11 line 23.; Figure 4, 404 No arrow. Specification page 13, lines 21-22.; Figure 5 at 510 No arrow. Specification: page 14, line 18 to page 15, line 9.)**, wherein said second connection is different than said first connection** (Figure 2. Specification page 8 line 3 to page 9 line 17.; Figure 3. Specification page 10 line 15 to page 11 line 23.; Figure 4 at 408

HTTP. Specification page 13, lines 21-22. v. Figure 4 at 402 TCP port X. Specification: page 13, lines 3-4.)**;**

means for evaluating the second connection for a response from a remote system indicating a successful second connection (Figure 2. Specification page 8 line 3 to page 9 line 17.; Figure 3. Specification page 10 line 15 to page 11 line 23.; Figure 4, 410 Yes arrow. Specification page 12, line 14 to page 14 line 17)**;**

means for initiating a third connection (Figure 2. Specification page 8 line 3 to page 9 line 17.; Figure 3. Specification page 10 line 15 to page 11 line 23.; Figure 4, 412. Specification page 12, line 14 to page 14 line 17.) **to go through said firewall if a successful second connection is not established** (Figure 2. Specification page 8 line 3 to page 9 line 17.; Figure 3. Specification page 10 line 15 to page 11 line 23.; Figure 4, 410 No arrow. Specification page 12, line 14 to page 14 line 17)**, wherein said third connection is different than said second connection and said first connection** (Figure 2. Specification page 8 line 3 to page 9 line 17.; Figure 3. Specification page 10 line 15 to page 11 line 23.; Figure 4, 412 HTTP proxy. Specification page 12, line 14 to page 14 line 17. v. Figure 4 at 408 HTTP. Specification page 13, lines 21-22. v. Figure 4 at 402 TCP port X. Specification: page 13, lines 3-4.)**; and**

means for evaluating the third connection for a response from a remote system indicating a successful third connection (Figure 2. Specification page 8 line 3 to page 9 line 17.; Figure 3. Specification page 10 line 15 to page 11 line 23.; Figure 4, 414 Yes arrow. Specification page 12, line 14 to page 14 line 17).

//

DEPENDENT CLAIMS

Appellant has indicated below representative Figures/Specifications for the following dependent claims – not all are included so as to keep the claims brief.

2. The method of claim 1, wherein the first connection, the second connection, and the third connection (Figure 4, at 402 & 404, 408 & 410, and 412 & 414 respectively) **is selected from the group consisting of Transmission Control Protocol (TCP) connection, User Datagram Protocol (UDP) connection, hypertext transfer protocol (HTTP) connection, hypertext transfer protocol (HTTP) connection via a proxy connection, and Internet Control Message Protocol (ICMP) connection** (Figure 4. Specification: page 20, lines 6-20.)**.**

3. The method according to claim 2, wherein initiating a TCP connection comprises initiating a TCP connection to a predefined address and port (Specification: page 13, lines 3-4)**.**

4. The method according to claim 2, wherein initiating a HTTP connection comprises initiating a HTTP connection to a predefined address using port 80 (Specification: page 16, line 17; page 23, lines 1-2)**.**

5. The method according to claim 2, wherein initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address and port (Figure 5, Specification: page 14, line 18 through page 17, line 15)**.**

6. The method according to claim 5, wherein determining a likely proxy address and port further comprises packet sniffing (Figure 5 at 502, 506; Specification: page 14, line 18 through page 17, line 15)**.**

7. The method according to claim 6, wherein packet sniffing further comprises:

sampling packets (Specification page 14, lines 20-22) **;**

extracting information from the sampled packets (Specification page 14 line 24 to page 15, line 1)**; and**

building a database of likely proxy addresses and ports (Specification page 15, lines 22-23)**.**

8. The method according to claim 7, wherein extracting information from the sampled packets comprises extracting TCP port information (Specification page 18, line 12 to page 19, line 6.)**.**

9. The method according to claim 7, wherein extracting information from the sampled packets comprises examining TCP packets for HTTP data (Specification page 19, lines 2-3.)**.**

10. The method of claim 2 further comprising using Internet Protocol (IP) (Specification page 20 lines 6-20.)**.**

11. The method according to claim 10, wherein initiating a HTTP connection via a proxy connection (Figure 5 at 508) **further comprises determining a likely proxy address by sampling packets and extracting IP addresses** (Specification page 18 line 3 to page 19 line 16.)**.**

12. The method of claim 2 further comprising using Ethernet with the Transmission Control Protocol (TCP) (Specification page 18 line 15 to page 19 line 6, page 24 lines 5-6.)**.**

13. The method according to claim 12, wherein initiating a HTTP connection via a proxy connection (Figure 5 at 508) **further comprises determining a likely proxy address by sampling packets and extracting Ethernet addresses** (Specification page 18 line 3 to page 19 line 16.)**.**

15. The machine-readable medium according to claim 14, further configuring said processor to perform the following:

implement the first connection (Figure 4 at 402)**, the second connection** (Figure 4 at 408)**, and the third connection** (Figure 4 at 412) **selected from the group consisting of Transmission Control Protocol (TCP) connection, User Datagram Protocol (UDP) connection, hypertext transfer protocol (HTTP) connection, hypertext transfer protocol (HTTP) proxy connection, and Internet Control Message Protocol (ICMP) connection** (Figure 4. Specification: page 20, lines 6-20.)**.**

16. The machine-readable medium according to claim 15, further configuring said processor to perform the following:

examine network traffic (Figure 6 at 608, Specification page 18 line 3 to page 19 line 16)**; and**

build a database (Figure 6 at 610) **of parameters likely to allow establishment of a HTTP connection via a proxy connection** (Specification page 18 line 3 to page 19 line 16)**.**

18. The system of claim 17, wherein the packet examining subsystem extracts port information (Specification page 18 line 3 to page 19 line 16).

19. The system of claim 18, wherein the packet examining subsystem extracts the port information based upon examining packet data content (Specification page 18 line 3 to page 19 line 16).

20. The system of claim 17, wherein the packet examining subsystem extracts address information (Specification page 18 line 3 to page 19 line 16).

21. The system of claim 20, wherein the packet examining subsystem extracts the address information based upon examining packet data content (Specification page 18 line 3 to page 19 line 16).

23. The apparatus of claim 22, wherein means for initiating the first connection, means for initiating the second connection, and means for initiating the third connection further comprises means for initiating a connection selected from the group consisting of Transmission Control Protocol (TCP) connection, User Datagram Protocol (UDP) connection, hypertext transfer protocol (HTTP) connection, hypertext transfer protocol (HTTP) proxy connection, and Internet Control Message Protocol (ICMP) connection (Figure 4. Specification: page 20, lines 6-20.).

24. The apparatus of claim 23, wherein means for initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address by sniffing packets and extracting information from the packets (Figure 5, Figure 6: Specification page 18 line 3 to page 19 line 16).

25. The apparatus of claim 23, wherein means for initiating a HTTP connection via a proxy connection (Figure 5 at 508) **further comprises determining a likely proxy address by receiving information from a computer connected to the firewall** (Specification page 16 line 8 to page 17 line 15)**.**

26. The apparatus of claim 22, further comprising means for updating firewall traversal strategies (Figure 5 at 512, Figure 6 at 610: Specification page 18 line 3 to page 19 line 16)**.**

//

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The issues presented on appeal are whether Applicant's claims 1-26 for the reasons stated in the Status of the Claims are unpatentable.

First

Claims 1-16 and 22-26 stand rejected under 35 U.S.C. § 112 as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Second

Claims 1-4, 10, 12, 14-16 and 22-26 stand rejected under 35 U.S.C. § 102(b) as being anticipated by Vellanki et al U.S. Patent No. 5,999,979.

Third

Claim 4 stands rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 1 above, and further in view of Bhide et al U.S. Patent No. 5,852,717.

Fourth

Claims 5-9, 11 and 13 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 1 above, and further in view of Fuh et al U.S. Patent No. 6,609,154 B1.

Fifth

Claim 16 stands rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 14 above, and further in view of Linden et al U.S. Patent No. 6,549,773 B1.

Sixth

Claims 24 and 25 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 22 above, and further in view of Fuh et al U.S. Patent No. 6,609,154 B1.

Seventh

Claim 26 is rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 22 above, and further in view of Montenegro U.S. Patent No. 6,233,688 B1.

//

VII. ARGUMENT

For the benefit of the board, the claims and the historical prosecution history of the instant application are illustrated on the following page. Claim structure is illustrated by indentation of the claims. Leftmost claims are independent. (I.e. claim 2 is dependent on claim 1. Claim 3 is dependent on 2, as is claim 4 and claim 5. Claim 6 is dependent on claim 5 which is dependent on 2, which is dependent on 1.)

Each claim is addressed below, however, Claim 1 is considered particularly relevant because the dependent claims and other independent claims and dependent claims are variations on claim 1.

Digi P001 Appeal historical claim tree.doc

CLAIM #	4th OA Rejection	3rd OA Rejection	2nd OA Rejection	1st OA Rejection
1	§112¶1 failure to comply with written description requirement §102(b) Vellanki	102(e) Purtell	102(e) Crump	101?, 102(b) Vellanki
2	§112¶1 failure to comply with written description requirement §102(b) Vellanki	102(e) Purtell	102(e) Crump	102(b) Vellanki
3	§112¶1 failure to comply with written description requirement §102(b) Vellanki	102(e) Purtell	102(e) Crump	102(b) Vellanki
4	§112¶1 failure to comply with written description requirement §102(b) Vellanki §103(a) Vellanki as applied to claim 1 above, ifvo Bhide	102(e) Purtell 103(a) Purtell as applied to claim 1 above, ifvo Bhide	102(e) Crump, 103(a) Crump ivo Bhide	102(b) Vellanki
5	§112¶1 failure to comply with written description requirement §103(a) Vellanki as applied to claim 1 above, ifvo Fuh	103(a) Purtell as applied to claim 1 above, ifvo Fuh	103(a) Crump ivo Fuh	102(b) Vellanki
6	§112¶1 failure to comply with written description requirement §103(a) Vellanki as applied to claim 1 above, ifvo Fuh	103(a) Purtell as applied to claim 1 above, ifvo Fuh	103(a) Crump ivo Fuh	103(a) Vellanki ivo Harvey
7	§112¶1 failure to comply with written description requirement §103(a) Vellanki as applied to claim 1 above, ifvo Fuh	103(a) Purtell as applied to claim 1 above, ifvo Fuh	103(a) Crump ivo Fuh	103(a) Vellanki ivo Harvey
8	§112¶1 failure to comply with written description requirement §103(a) Vellanki as applied to claim 1 above, ifvo Fuh	103(a) Purtell as applied to claim 1 above, ifvo Fuh	103(a) Crump ivo Fuh	103(a) Vellanki ivo Harvey
9	§112¶1 failure to comply with written description requirement §103(a) Vellanki as applied to claim 1 above, ifvo Fuh	103(a) Purtell as applied to claim 1 above, ifvo Fuh	103(a) Crump ivo Fuh	103(a) Vellanki ivo Harvey
10	§112¶1 failure to comply with written description requirement §102(b) Vellanki	102(e) Purtell	102(e) Crump	102(b) Vellanki

CLAIM #	4th OA Rejection	3rd OA Rejection	2nd OA Rejection	1st OA Rejection
11	§112¶1 failure to comply with written description requirement §103(a) Vellanki as applied to claim 1 above, ifvo Fuh	103(a) Purtell as applied to claim 1 above, ifvo Fuh	103(a) Crump ivo Fuh	102(b) Vellanki
12	§112¶1 failure to comply with written description requirement §102(b) Vellanki	102(e) Purtell	102(e) Crump	103(a) Vellanki ivo Cunningham
13	§112¶1 failure to comply with written description requirement §103(a) Vellanki as applied to claim 1 above, ifvo Fuh	103(a) Purtell as applied to claim 1 above, ifvo Fuh	103(a) Crump ivo Fuh	103(a) Vellanki ivo Cunningham
14	§112¶1 failure to comply with written description requirement §102(b) Vellanki	102(e) Purtell	102(e) Crump	102(b) Vellanki
15	§112¶1 failure to comply with written description requirement §102(b) Vellanki	102(e) Purtell	102(e) Crump	102(b) Vellanki
16	§112¶1 failure to comply with written description requirement §102(b) Vellanki §103(a) Vellanki as applied to claim 1 above, ifvo Linden	102(e) Purtell 103(a) Purtell as applied to claim 14 above, ifvo Linden	102(e) Crump, 103(a) Crump ivo Linden	103(a) Vellanki ivo Harvey
17	§102(b) Vellanki	102(b) Freund	102(e) Qu	101?, 102(a) Harvey
18	§102(b) Vellanki	102(b) Freund	102(e) Qu	102(a) Harvey
19	§102(b) Vellanki	102(b) Freund	102(e) Qu	102(a) Harvey
20	§102(b) Vellanki	102(b) Freund	102(e) Qu	102(a) Harvey
21	§102(b) Vellanki	102(b) Freund	102(e) Qu	102(a) Harvey
22	§112¶1 failure to comply with written description requirement §102(b) Vellanki	102(e) Purtell	102(e) Crump	101?, 102(b) Vellanki
23	§112¶1 failure to comply with written description requirement §102(b) Vellanki	102(e) Purtell	102(e) Crump	102(b) Vellanki
24	§112¶1 failure to comply with written description requirement §102(b) Vellanki §103(a) Vellanki as applied to claim 22 above, ifvo Fuh	102(e) Purtell 103(a) Purtell as applied to claim 22 above, ifvo Fuh	102(e) Crump, 103(a) Crump ivo Fuh	103(a) Vellanki ivo Harvey

CLAIM #	**4th OA Rejection**	**3rd OA Rejection**	**2nd OA Rejection**	**1st OA Rejection**
25	§112¶1 failure to comply with written description requirement §102(b) Vellanki §103(a) Vellanki as applied to claim 22 above, ifvo Fuh	102(e) Purtell 103(a) Purtell as applied to claim 22 above, ifvo Fuh	102(e) Crump, 103(a) Crump ivo Fuh	102(b) Vellanki
26	§112¶1 failure to comply with written description requirement §102(b) Vellanki §103(a) Vellanki as applied to claim 22 above, ifvo Montenegro	102(e) Purtell 103(a) Purtell as applied to claim 22 above, ifvo Montenegro	102(e) Crump, 103(a) Crump ivo Montenegro	102(b) Vellanki

INTRODUCTION

The present invention relates to a method and apparatus for traversing a firewall. Many devices, such as an air conditioner, a washer, a dryer, or other appliances would benefit from network connectivity (see Applicant's Figure 3).

A device located behind a firewall is presented with challenges in attempting to contact an external or outside resource. Likewise, an external device attempting to reach an internal resource behind a firewall is presented with the need to get through the firewall. When the devices behind the firewall are computers with keyboards, monitors, and loadable software, it is often possible to pull up configuration screens to properly configure the device for communication through the firewall. It is not so easy for an appliance type device that may be lacking user input capability to be configured to get through a firewall. This presents a problem.

Applicant's invention is directed to a variety of techniques for solving this problem.

//

Claims 1-16 and 22-26 Rejection under 35 U.S.C. § 112 ¶1

The Office at 6 states:

> 6. Claims 1-16 and 22-26 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.
>
> The applicant has amended independent claims 1, 14 and 22 to include the limitations "wherein said second connection is different than said first connection" and "wherein said third connection is different than said second connection and said first connection". After a careful review of the specification, the examiner finds no support in the specification for these limitations. If the applicant finds this an error, the examiner invites the applicant to point out the support in the specification.

Applicant submits that as detailed above each and every element of claims 1-16 and 22-26 are detailed in the specification as initially filed both in Figures and the written description.

The Examiner only points with particularity to claims 1, 14, and 22 with respect to sequence limitations.

Firstly, Applicant submits that Figure 4 for example, as originally filed clearly shows a first TCP connection, a second HTTP connection, and a third HTTP connection via proxy. One of skill in the art at the time of the invention clearly knows that these are not the same. Thus Applicant has shown support for the amended limitation of "wherein said second connection is different than said first connection" and "wherein said third connection is different than said second connection and said first connection".

Secondly, additional support may be found in the Specification at page 13 line 3 to page 14 line 6. Specifically in the cited section "Referring to Figure 4 … tries to open a TCP connection to a prespecified port (denoted as x) … If the TCP port x connection is not established then the device tries to open a HTTP connection 408. If the HTTP connection is not established then the device tries to open a HTTP connection via a proxy connection 412." and additionally is disclosed "If the HTTP connection via a proxy connection is not established then the device may try other options to open a connection 416, such as trying a different address with the sequence described above." [Emphasis added.]

Thirdly, the Examiner's hand waiving without a reasoned explanation as to why the other claims fail to satisfy 35 U.S.C. § 112 ¶1 is not a good faith effort at, and does not advance, prosecution. Additionally, to raise a 112 rejection for the first time after 3 prior examinations does not show diligent examination.

Applicant submits that the claims satisfy 35 U.S.C. § 112 ¶1 and respectfully requests this Board to remove this rejection and allow the claims.

//

Claims 1-3, 10, 12, 14, 15, 17-23 Rejection under 35 U.S.C. § 102(b)

The Office at 7 states:

7. Claims 1-4, 10, 12, 14-16 and 22-26 are rejected under 35 U.S.C. 102(b) as being anticipated by Vellanki et al U.S. Patent No. 5,999,979.[†]

(Emphasis in original.)

[†] Applicant notes for the Board that the Examiner's numbering is not consistent with actual arguments presented (probably as a result of cut and paste operations).

Claims 1, 14, and 22 Rejection under 35 U.S.C. § 102(b) - Vellanki

The Office at 7 states: (continued)

As to claims 1, 14 and 22, Vellanki et al discloses a method for traversing a firewall, comprising:

> initiating a first connection to go through the firewall [column 6, lines 1-48];
>
> evaluating the first connection for a response from a remote system indicating a successful first connection [column 6, lines 1-48];
>
> initiating a second connection to go through the firewall if a successful first connection is not established, wherein the second connection is different than the first connection [column 6, lines 1-48];
>
> evaluating the second connection for a response from a remote system indicating a successful second connection [column 6, lines 1-48];
>
> initiating a third connection to go through the firewall if a successful second connection is not established, wherein the third connection is different that the second connection and the first connection [column 6, lines 1-48]; and
>
> evaluating the third connection for a response from a remote system indicating a successful third connection [column 11, lines 3-9].

The cited references state in part:

[column 6, lines 1-48] - Vellanki

In accordance with one particular advantageous embodiment, the inventive autodetect mechanism simultaneously employs multiple threads, through multiple connections, to initiate communication with the server computer, e.g., server 104. Each thread preferably employs a different protocol and requests the server computer to respond to the client computer using the protocol associated with that thread. For example, client computer 106 may, employing the autodetect mechanism, initiate five different threads, using respectively the TCP, UDP, one of HTTP and HTTP proxy, HTTP through port (multiplex) 80, and HTTP through port (multiplex) 8080 protocols to request server 104 to respond.

Upon receiving a request, server 104 responds with data using the same protocol as that associated with the thread on which the request arrives. If one or more protocols is blocked and fails to reach server 104 (e.g., by a firewall), no response employing the blocked protocol would of course be transmitted from server 104 to client computer 106. Further, some of the protocols transmitted from server 104 to client computer 106 may be blocked as well. Accordingly, client computer may receive only a subset of the responses sent from server 104.

In one embodiment, client computer 106 monitors the set of received responses. If the predefined "best" protocol is received, that protocol is then selected for communication by client computer 106. The predefined "best" protocol may be defined in advance by the user and/or the application program. If the predefined "best" protocol is, however, blocked (as the request is transmitted from the client computer or as the response is transmitted from the server, for example) the most advantageous protocol may simply be selected from the set of protocols received back by the client computer. In one embodiment, the selection may be made among the set of protocols received back by the client computer within a predefined time period after the requests are sent out in parallel.

The selection of the most advantageous protocol for communication among the protocols received by client computer 106 may be performed in accordance with some predefined priority. For example, in the real-time data rendering application, the UDP protocol may be preferred over TCP protocol, which may be in turned preferred over the HTTP protocol. This is because UDP protocol typically can handle a greater data transmission rate and may allow client computer 106 to exercise a greater degree of control over the transmission of data packets.

[column 11, lines 3-9] - Vellanki

When a server gets an HTTP request, it answers the client with an HTTP Response. Responses are typically composed of a Status-Line, one or more headers, and an Entity-Body. In one embodiment of this invention, the response to the media commands is sent as the Entity-Body of the response to the original HTTP request that carried the media command.

Applicant's claims 1, 14, and 22 recite:

1. A method for traversing a firewall, comprising:

initiating a first connection to go through said firewall;

evaluating the first connection for a response from a remote system indicating a successful first connection;

initiating a second connection to go through said firewall if a successful first connection is not established, wherein said second connection is different than said first connection;

evaluating the second connection for a response from a remote system indicating a successful second connection;

initiating a third connection to go through said firewall if a successful second connection is not established, wherein said third connection is different than said second connection and said first connection; and

evaluating the third connection for a response from a remote system indicating a successful third connection.

14. A machine-readable medium having stored thereon instructions, which when executed by a processor, causes said processor to perform the following:

initiate a first connection to go through a firewall;

evaluate the first connection for a response from a remote system indicating a successful first connection;

initiate a second connection to go through said firewall if a successful first connection is not established, wherein said second connection is different than said first connection;

evaluate the second connection for a response from a remote system indicating a successful second connection;

initiate a third connection to go through said firewall if a successful second connection is not established, wherein said third connection is different than said second connection and said first connection; and

evaluate the third connection for a response from a remote system indicating a successful third connection.

22. A method for traversing a firewall, comprising:

means for initiating a first connection to go through said firewall;

means for evaluating the first connection for a response from a remote system indicating a successful first connection;

means for initiating a second connection to go through said firewall if a successful first connection is not established, wherein said second connection is different than said first connection;

means for evaluating the second connection for a response from a remote system indicating a successful second connection;

means for initiating a third connection to go through said firewall if a successful second connection is not established, wherein said third connection is different than said second connection and said first connection; and

means for evaluating the third connection for a response from a remote system indicating a successful third connection.

Claim 1 Rejection under 35 U.S.C. § 102(b) - Vellanki

Applicant's claim 1 recites:

1. A method for traversing a firewall, comprising:

initiating a first connection to go through said firewall;

evaluating the first connection for a response from a remote system indicating a successful first connection;

initiating a second connection to go through said firewall **if** a successful first connection is not established, wherein said second connection is different than said first connection;

evaluating the second connection for a response from a remote system indicating a successful second connection;

initiating a third connection to go through said firewall **if** a successful second connection is not established, wherein said third connection is different than said second connection and said first connection; and

evaluating the third connection for a response from a remote system indicating a successful third connection.

[Emphases added.]

Vellanki et al ("Vellanki") specifically discloses at the cited reference "…simultaneously employs multiple threads, through multiple connections, to initiate communication with the server computer". [Emphasis added.]

Furthermore, Vellanki in Figure 7 and the accompanying text (col 13, lines 56-67) clearly states:

> In step 704, the client begins the autodetect sequence by **starting in parallel** the control thread 794, along with five protocol threads 790, 792, 796, 798, and 788. As the term is used herein, **parallel refers to both** the situation wherein the multiple protocol threads are sent parallely **starting at substantially the same time** (having substantially similar starting time), **and** the situation wherein the multiple protocol **threads simultaneously execute** (executing at the same time), irrespective when each protocol thread is initiated. In the latter case, the multiple threads may have, for example, staggered start time and the initiation of one thread may not depend on the termination of another thread.
>
> [Emphases added.]

Applicant's claim limitation has a second or third connection initiated only if the prior one fails. Thus there is a fundamental difference between what Applicant has claimed in claim 1 and what Vellanki discloses. **Applicant's method is sequential whereas Vellanki's method is parallel.**

Since Vellanki does not disclose this limitation of Applicant's claim 1, Vellanki does not anticipate Applicant's claim 1. Applicant therefore respectfully requests allowance of claim 1 and all claims dependent on claim 1.

Claim 14 and Claim 22 Rejection under 35 U.S.C. § 102(b) - Vellanki

Applicant's claim 14 is a Beauregard style version of claim 1. Thus, for the same reasons as in the Claim 1 discussion above, Vellanki does not anticipate Applicant's claim 14. Applicant requests allowance of claim 14 and all claims dependent on claim 14.

Applicant's claim 22 is a means plus function style version of claim 1. Thus, for the same reasons as the Claim 1 discussion above, Vellanki does not anticipate Applicant's claim 22. Applicant requests allowance of claim 22 and all claims dependent on claim 22.

//

Claims 2, 15 and 23 Rejection under 35 U.S.C. § 102(b) - Vellanki

The Office at 7 states: (continued)

As to claims 2, 15 and 23, Vellanki et al discloses that the first connection, the second connection, and the third connection is selected from the group consisting of Transmission Control Protocol (TCP) connection, User Datagram Protocol (UDP) connection, hypertext transfer protocol (HTTP) connection, hypertext transfer protocol (HTTP) connection via a proxy connection, and Internet Control Message Protocol (ICMP) connection [column 6, lines 1-48].

The cited reference states in part:

[column 6, lines 1-48] - Vellanki

In accordance with one particular advantageous embodiment, the inventive autodetect mechanism simultaneously employs multiple threads, through multiple connections, to initiate communication with the server computer, e.g., server 104. Each thread preferably employs a different protocol and requests the server computer to respond to the client computer using the protocol associated with that thread. For example, client computer 106 may, employing the autodetect mechanism, initiate five different threads, using respectively the TCP, UDP, one of HTTP and HTTP proxy, HTTP through port (multiplex) 80, and HTTP through port (multiplex) 8080 protocols to request server 104 to respond.

Upon receiving a request, server 104 responds with data using the same protocol as that associated with the thread on which the request arrives. If one or more protocols is blocked and fails to reach server 104 (e.g., by a firewall), no response employing the blocked protocol would of course be transmitted from server 104 to client computer 106. Further, some of the protocols transmitted from server 104 to client computer 106 may be blocked as well. Accordingly, client computer may receive only a subset of the responses sent from server 104.

In one embodiment, client computer 106 monitors the set of received responses. If the predefined "best" protocol is received, that protocol is then selected for communication by client computer 106. The predefined "best" protocol may be defined in advance by the user and/or the application program. If the predefined "best" protocol is, however, blocked (as the request is transmitted from the client computer or as the response is transmitted from the server, for example) the most advantageous protocol may simply be selected from the set of protocols received back by the client computer. In one embodiment, the selection may be made among the set of protocols received back by the client computer within a predefined time period after the requests are sent out in parallel.

The selection of the most advantageous protocol for communication among the protocols received by client computer 106 may be performed in accordance with some predefined priority. For example, in the real-time data rendering application, the UDP protocol may be preferred over TCP protocol, which may be in turned preferred over the HTTP protocol. This is because UDP protocol typically can handle a greater data transmission rate and may allow client computer 106 to exercise a greater degree of control over the transmission of data packets.

Applicant's claims 2, 15, and 23 recite:

2. The method of claim 1, wherein the first connection, the second connection, and the third connection is selected from the group consisting of Transmission Control Protocol (TCP) connection, User Datagram Protocol (UDP) connection, hypertext transfer protocol (HTTP)

connection, hypertext transfer protocol (HTTP) connection via a proxy connection, and Internet Control Message Protocol (ICMP) connection.

15. The machine-readable medium according to claim 14, further configuring said processor to perform the following:

implement the first connection, the second connection, and the third connection selected from the group consisting of Transmission Control Protocol (TCP) connection, User Datagram Protocol (UDP) connection, hypertext transfer protocol (HTTP) connection, hypertext transfer protocol (HTTP) proxy connection, and Internet Control Message Protocol (ICMP) connection.

23. The apparatus of claim 22, wherein means for initiating the first connection, means for initiating the second connection, and means for initiating the third connection further comprises means for initiating a connection selected from the group consisting of Transmission Control Protocol (TCP) connection, User Datagram Protocol (UDP) connection, hypertext transfer protocol (HTTP) connection, hypertext transfer protocol (HTTP) proxy connection, and Internet Control Message Protocol (ICMP) connection.

Applicant submits that claim 2, claim 15, and claim 23 are dependent on independent claim 1, claim 14, and claim 22 respectively and as such inherit the limitations of their respective independent claim. As discussed above Vellanki fails to disclose limitations in independent claim 1, claim 14, and claim 22 and therefore Vellanki cannot anticipate

Applicant's claim 2, claim 15, and claim 23. Further, Vellanki cannot anticipate the further limitation of Applicant's claim 2, claim 15, and claim 23 of a first, second, and third connection selected from TCP, UDP, HTTP, HTTP proxy connection, and ICMP connection. Applicant respectfully requests allowance of claim 2, claim 15, and claim 23.

Claim 3 Rejection under 35 U.S.C. § 102(b) - Vellanki

The Office at 7 states: (continued)

As to claim 3, Vellanki et al discloses that initiating a TCP connection comprises initiating a TCP connection to a predefined address and port [column 6, lines 1-48].

The cited reference states in part:

[column 6, lines 1-48] - Vellanki

In accordance with one particular advantageous embodiment, the inventive autodetect mechanism simultaneously employs multiple threads, through multiple connections, to initiate communication with the server computer, e.g., server 104. Each thread preferably employs a different protocol and requests the server computer to respond to the client computer using the protocol associated with that thread. For example, client computer 106 may, employing the autodetect mechanism, initiate five different threads, using respectively the TCP, UDP, one of HTTP and HTTP proxy, HTTP through port (multiplex) 80, and HTTP through port (multiplex) 8080 protocols to request server 104 to respond.

Upon receiving a request, server 104 responds with data using the same protocol as that associated with the thread on which the request arrives. If one or more protocols is blocked and fails to reach server 104 (e.g., by a firewall), no response employing the blocked protocol would of course be transmitted from server 104 to client computer 106. Further, some of the protocols transmitted from server 104 to client computer 106 may be blocked as well. Accordingly, client computer may receive only a subset of the responses sent from server 104.

In one embodiment, client computer 106 monitors the set of received responses. If the predefined "best" protocol is received, that protocol is then selected for communication by client computer 106. The predefined "best" protocol may be defined in advance by the user and/or the application program. If the predefined "best" protocol is, however, blocked (as the request is transmitted from the client computer or as the response is transmitted from the server, for example) the most advantageous protocol may simply be selected from the set of protocols received back by the client computer. In one embodiment, the selection may be made among the set of protocols received back by the client computer within a predefined time period after the requests are sent out in parallel.

The selection of the most advantageous protocol for communication among the protocols received by client computer 106 may be performed in accordance with some predefined priority. For example, in the real-time data rendering application, the UDP protocol may be preferred over TCP protocol, which may be in turned preferred over the HTTP protocol. This is because UDP protocol typically can handle a greater data transmission rate and may allow client computer 106 to exercise a greater degree of control over the transmission of data packets.

Applicant's claim 3 recites:

3. The method according to claim 2, wherein initiating a TCP connection comprises initiating a TCP connection to a predefined address and port.

Applicant submits that Claim 3 is dependent on claim 2, which is dependent on claim 1 and as discussed above in the claim 1 discussion, Vellanki fails to disclose a limitation of claim 1, therefore Vellanki cannot anticipate Applicant's claim 3. Further, Vellanki cannot anticipate the further limitation of Applicant's claim 3. Applicant respectfully requests allowance of claim 3.

Claim 10 Rejection under 35 U.S.C. § 102(b) - Vellanki

The Office at 7 states: (continued)

As to claim 10, Vellanki et al discloses using Internet Protocol (IP) [column 6, lines 1-48].

The cited reference states in part:

[column 6, lines 1-48] - Vellanki

In accordance with one particular advantageous embodiment, the inventive autodetect mechanism simultaneously employs multiple threads, through multiple connections, to initiate communication with the server computer, e.g., server 104. Each thread preferably employs a different protocol and requests the server computer to respond to the client computer using the protocol associated with that thread. For example, client computer 106 may, employing the autodetect mechanism, initiate five different threads, using respectively the TCP, UDP, one of HTTP and HTTP proxy, HTTP through port (multiplex) 80, and HTTP through port (multiplex) 8080 protocols to request server 104 to respond.

Upon receiving a request, server 104 responds with data using the same protocol as that associated with the thread on which the request arrives. If one or more protocols is blocked and fails to reach server 104 (e.g., by a firewall), no response employing the blocked protocol would of course be transmitted from server 104 to client computer 106. Further, some of the protocols transmitted from server 104 to client computer 106 may be blocked as well. Accordingly, client computer may receive only a subset of the responses sent from server 104.

In one embodiment, client computer 106 monitors the set of received responses. If the predefined "best" protocol is received, that protocol is then selected for communication by client computer 106. The predefined "best" protocol may be defined in advance by the user and/or the application program. If the predefined "best" protocol is, however, blocked (as the request is transmitted from the client computer or as the response is transmitted from the server, for example) the most advantageous protocol may simply be selected from the set of protocols received back by the client computer. In one embodiment, the selection may be made among the set of protocols received back by the client computer within a predefined time period after the requests are sent out in parallel.

The selection of the most advantageous protocol for communication among the protocols received by client computer 106 may be performed in accordance with some predefined priority. For example, in the real-time data rendering application, the UDP protocol may be preferred over TCP protocol, which may be in turned preferred over the HTTP protocol. This is because UDP protocol typically can handle a greater data transmission rate and may allow client computer 106 to exercise a greater degree of control over the transmission of data packets.

Applicant's claim 10 recites:

10. The method of claim 2 further comprising using Internet Protocol (IP).

Applicant submits that Claim 10 is dependent on claim 2, which is dependent on claim 1 and as discussed above in the claim 1 discussion, Vellanki fails to disclose a limitation of claim 1, therefore Vellanki cannot anticipate Applicant's claim 10. Further, Vellanki cannot anticipate the further limitation of Applicant's claim 10. Applicant respectfully requests allowance of claim 10.

Claim 12 Rejection under 35 U.S.C. § 102(b) - Vellanki

The Office at 7 states: (continued)

As to claim 12, Vellanki et al discloses using Ethernet with the Transmission Control Protocol (TCP) [column 6, lines 1-48].

The cited reference states in part:

[column 6, lines 1-48] - Vellanki

In accordance with one particular advantageous embodiment, the inventive autodetect mechanism simultaneously employs multiple threads, through multiple connections, to initiate communication with the server computer, e.g., server 104. Each thread preferably employs a different protocol and requests the server computer to respond to the client computer using the protocol associated with that thread. For example, client computer 106 may, employing the autodetect mechanism, initiate five different threads, using respectively the TCP, UDP, one of HTTP and HTTP proxy, HTTP through port (multiplex) 80, and HTTP through port (multiplex) 8080 protocols to request server 104 to respond.

Upon receiving a request, server 104 responds with data using the same protocol as that associated with the thread on which the request arrives. If one or more protocols is blocked and fails to reach server 104 (e.g., by a firewall), no response employing the blocked protocol would of course be transmitted from server 104 to client computer 106. Further, some of the protocols transmitted from server 104 to client computer 106 may be blocked as well. Accordingly, client computer may receive only a subset of the responses sent from server 104.

In one embodiment, client computer 106 monitors the set of received responses. If the predefined "best" protocol is received, that protocol is then selected for communication by client computer 106. The predefined "best" protocol may be defined in advance by the user and/or the application program. If the predefined "best" protocol is, however, blocked (as the request is transmitted from the client computer or as the response is transmitted from the server, for example) the most advantageous protocol may simply be selected from the set of protocols received back by the client computer. In one embodiment, the selection may be made among the set of protocols received back by the client computer within a predefined time period after the requests are sent out in parallel.

The selection of the most advantageous protocol for communication among the protocols received by client computer 106 may be performed in accordance with some predefined priority. For example, in the real-time data rendering application, the UDP protocol may be preferred over TCP protocol, which may be in turned preferred over the HTTP protocol. This is because UDP protocol typically can handle a greater data transmission rate and may allow client computer 106 to exercise a greater degree of control over the transmission of data packets.

Applicant's claim 12 recites:

12. The method of claim 2 further comprising using Ethernet with the Transmission Control Protocol (TCP).

Applicant submits that Claim 12 is dependent on claim 2, which is dependent on claim 1 and as discussed above in the claim 1 discussion, Vellanki fails to disclose a limitation of claim 1, therefore Vellanki cannot anticipate Applicant's claim 12. Further, Vellanki cannot anticipate the further limitation of Applicant's claim 12. Applicant respectfully requests allowance of claim 12.

//

Claim 17 Rejection under 35 U.S.C. § 102(b) - Vellanki

The Office at 7 states: (continued)

As to claim 17, Vellanki et al discloses a firewall traversal system comprising:

> a main system coupled to storage [column 5, lines 56-67];
>
> a communication subsystem coupled to the main system and a communication medium on one side of a firewall [column 11, lines 27-42];
>
> a packet examining subsystem coupled to the communication subsystem [column 11, lines 27-42]; and
>
> a database system coupled to the packet examining subsystem and the main system [column 11, lines 27-42].

The cited references state in part:

[column 5, lines 56-67] - Vellanki

In accordance with one aspect of the present invention, the client computer in a heterogeneous client-server computer network (e.g., client computer 106 in FIG. 1) is provided with an autodetect mechanism. When executed, the autodetect mechanism advantageously permits client computer 106 to select, in an efficient and automatic manner, the most advantageous protocol for communication between the client computer and its server. Once the most advantageous protocol is selected, parameters pertaining to the selected protocol are saved to enable the client computer, in future sessions, to employ the same selected protocol for communication.

[column 11, lines 27-42] - Vellanki

Some firewalls may be restrictive with respect to HTTP data and may permit HTTP packets to traverse only on a certain port, e.g., port 80 and/or port 8080. FIG. 5C illustrates one such situation. In this case, the control and data communications for the various data stream, e.g., audio, video, and/or annotation associated with different rendering sessions (and different clients) may be multiplexed using conventional multiplexer code and/or circuit 506 at client application 300 prior to being sent via port 502 (which may represent, for example, HTTP port 80 or HTTP port 8080). The inventive combined use of the HTTP protocol and of the multiplexer for transmitting media control and data is referred to as the HTTP multiplex protocol, and can be used to send this data across firewalls that only allow HTTP traffic on specific ports, e.g., port 80 or 8080.

Applicant's claim 17 recites:

17. A firewall traversal system comprising:

a main system coupled to storage;

a communication subsystem coupled to the main system and a communication medium on one side of a firewall;

a packet examining subsystem coupled to the communication subsystem; and

a database system coupled to the packet examining subsystem and the main system.

<u>Firstly</u>, Applicant submits that Vellanki at the cited reference for this element

```
a main system coupled to storage [column 5, lines 56-67];
```

is silent as to whether the "main system" is the client or the server as shown in Vellanki Fig. 1.

Reading further the second element also has a main system a communication subsystem coupled to the main system and a communication medium on one side of a firewall [column 11, lines 27-42];

and from that, as shown in Vellanki Fig. 5C we still have no clue as to whether the "main system" is the client or the server.

However, neither the server nor the client in Fig. 5C has Applicant's limitation of "a communication subsystem coupled to the main system and a communication medium <u>on one side of a firewall."</u> [Emphasis added.] That is, Vellanki does not disclose the limitation of a communication subsystem coupled to a main system and a communication medium on ONE side of a firewall. Vellanki shows no communication subsystem or it being coupled to a main system and to a communication medium on one side of a firewall. Fig. 5C of Vellanki shows only a server and client communicating through a firewall.

Since Vellanki does not show this limitation it cannot anticipate Applicant's claim 17.

<u>Secondly</u>, with regard to this element:

a packet examining subsystem coupled to the communication subsystem [column 11, lines 27-42]

Vellanki as explained above does not disclose a communication subsystem.

<u>Thirdly</u>, with regard to this element:

a packet examining subsystem coupled to the communication subsystem [column 11, lines 27-42]

Vellanki does not disclose any packet examining subsystem.

Fourthly, with regard to this element:

> a database system coupled to the packet examining subsystem and the main system [column 11, lines 27-42].

Vellanki does not disclose any database system, nor does Vellanki disclose any database system coupled to a packet examining subsystem, nor does Vellanki disclose any database system coupled to a packet examining subsystem and the main system.

For the multiple reasons cited above Vellanki does not show several limitations of Applicant's claim 17, therefore it cannot anticipate Applicant's claim 17. Applicant respectfully requests allowance of claim 17.

//

Claim 18 Rejection under 35 U.S.C. § 102(b) - Vellanki

The Office at 7 states: (continued)

As to claim 18, Vellanki et al discloses that the packet examining subsystem extracts port information [column 11, lines 27-42].

The cited reference states in part:

[column 11, lines 27-42] - Vellanki

Some firewalls may be restrictive with respect to HTTP data and may permit HTTP packets to traverse only on a certain port, e.g., port 80 and/or port 8080. FIG. 5C illustrates one such situation. In this case, the control and data communications for the various data stream, e.g., audio, video, and/or annotation associated with different rendering sessions (and different clients) may be multiplexed using conventional multiplexer code and/or circuit 506 at client application 300 prior to being sent via port 502 (which may represent, for example, HTTP port 80 or HTTP port 8080). The inventive combined use of the HTTP protocol and of the multiplexer for transmitting media control and data is referred to as the HTTP multiplex protocol, and can be used to send this data across firewalls that only allow HTTP traffic on specific ports, e.g., port 80 or 8080.

Applicant's claim 18 recites:

18. The system of claim 17, wherein the packet examining subsystem extracts port information.

Firstly, Claim 18 is dependent on claim 17, and as discussed above in the claim 17 discussion, Vellanki fails to disclose several limitations of claim 17, therefore Vellanki cannot anticipate Applicant's claim 18.

Secondly, with respect to the further limitation wherein

the packet examining subsystem extracts port information [column 11, lines 27-42]

Vellanki does not disclose extracting port information. First there is no packet examining subsystem and second there is no port information extracted. Vellanki is using "...a multiplexer for transmitting media control and data...".

For the above reasons, Vellanki cannot anticipate Applicant's claim 18. Applicant respectfully requests allowance of claim 18.

Claim 19 Rejection under 35 U.S.C. § 102(b) - Vellanki

The Office at 7 states: (continued)

As to claim 19, Vellanki et al discloses that the packet examining subsystem extracts the port information based upon examining packet data content [column 11, lines 27-42].

The cited reference states in part:

[column 11, lines 27-42] - Vellanki

Some firewalls may be restrictive with respect to HTTP data and may permit HTTP packets to traverse only on a certain port, e.g., port 80 and/or port 8080. FIG. 5C illustrates one such situation. In this case, the control and data communications for the various data stream, e.g., audio, video, and/or annotation associated with different rendering sessions (and different clients) may be multiplexed using conventional multiplexer code and/or circuit 506 at client application 300 prior to being sent via port 502 (which may represent, for example, HTTP port 80 or HTTP port 8080). The inventive combined use of the HTTP protocol and of the multiplexer for transmitting media control and data is referred to as the HTTP multiplex protocol, and can be used to send this data across firewalls that only allow HTTP traffic on specific ports, e.g., port 80 or 8080.

Applicant's claim 19 recites:

19. The system of claim 18, wherein the packet examining subsystem extracts the port information based upon examining packet data content.

<u>Firstly</u>, Claim 19 is dependent on claim 18, which is dependent on claim 17 and as discussed above in the claim 17 and claim 18 discussion, Vellanki fails to disclose several limitations of claim 17 and claim 18, therefore Vellanki cannot anticipate Applicant's claim 19.

<u>Secondly</u>, with respect to the further limitation wherein

the packet examining subsystem extracts the port information based upon examining packet data content [column 11, lines 27-42].

Vellanki does not disclose extracting port information, nor does it examine packet data content. First there is no packet examining subsystem, second there is no port information extracted, and third there is no examining packet data content.

For the above reasons, Vellanki cannot, and does not, anticipate Applicant's claim 19. Applicant respectfully requests allowance of claim 19.

Claim 20 Rejection under 35 U.S.C. § 102(b) - Vellanki

The Office at 7 states: (continued)

As to claim 20, Vellanki et al discloses that the packet examining subsystem extracts address information [column 11, lines 27-42].

The cited reference states in part:

[column 11, lines 27-42] - Vellanki

Some firewalls may be restrictive with respect to HTTP data and may permit HTTP packets to traverse only on a certain port, e.g., port 80 and/or port 8080. FIG. 5C illustrates one such situation. In this case, the control and data communications for the various data stream, e.g., audio, video, and/or annotation associated with different rendering sessions (and different clients) may be multiplexed using conventional multiplexer code and/or circuit 506 at client application 300 prior to being sent via port 502 (which may represent, for example, HTTP port 80 or HTTP port 8080). The inventive combined use of the HTTP protocol and of the multiplexer for transmitting media control and data is referred to as the HTTP multiplex protocol, and can be used to send this data across firewalls that only allow HTTP traffic on specific ports, e.g., port 80 or 8080.

Applicant's claim 20 recites:

20. The system of claim 17, wherein the packet examining subsystem extracts address information.

Firstly, Claim 20 is dependent on claim 17, and as discussed above in the claim 17 discussion, Vellanki fails to disclose several limitations of claim 17, therefore Vellanki cannot anticipate Applicant's claim 20.

Secondly, with respect to the further limitation wherein

the packet examining subsystem extracts address information [column 11, lines 27-42].

Vellanki does not disclose extracting address information. First there is no packet examining subsystem and second there is no address information extracted. Vellanki is using "...a multiplexer for transmitting media control and data…".

For the above reasons, Vellanki cannot anticipate Applicant's claim 20. Applicant respectfully requests allowance of claim 20.

Claim 21 Rejection under 35 U.S.C. § 102(b) - Vellanki

The Office at 7 states: (continued)

As to claim 21, Vellanki et al discloses that the packet examining subsystem extracts the address information based upon examining packet data content [column 11, lines 27-42].

The cited reference states in part:

[column 11, lines 27-42] - Vellanki

Some firewalls may be restrictive with respect to HTTP data and may permit HTTP packets to traverse only on a certain port, e.g., port 80 and/or port 8080. FIG. 5C illustrates one such situation. In this case, the control and data communications for the various data stream, e.g., audio, video, and/or annotation associated with different rendering sessions (and different clients) may be multiplexed using conventional multiplexer code and/or circuit 506 at client application 300 prior to being sent via port 502 (which may represent, for example, HTTP port 80 or HTTP port 8080). The inventive combined use of the HTTP protocol and of the multiplexer for transmitting media control and data is referred to as the HTTP multiplex protocol, and can be used to send this data across firewalls that only allow HTTP traffic on specific ports, e.g., port 80 or 8080.

Applicant's claim 21 recites:

21. The system of claim 20, wherein the packet examining subsystem extracts the address information based upon examining packet data content.

Firstly, Claim 21 is dependent on claim 20, which is dependent on claim 17 and as discussed above in the claim 17 and claim 20 discussion, Vellanki fails to disclose several limitations of claim 17 and claim 20, therefore Vellanki cannot anticipate Applicant's claim 19.

Secondly, with respect to the further limitation wherein

the packet examining subsystem extracts the address information based upon examining packet data content [column 11, lines 27-42]

Vellanki does not extract address information, nor does it examine packet data content. First there is no packet examining subsystem, second there is no address information extracted, and third there is no examining packet data content.

For the above reasons, Vellanki cannot, and does not, anticipate Applicant's claim 21. Applicant respectfully requests allowance of claim 21.

<u>Claim 4 Rejection under 35 U.S.C. § 103(a) – Vellanki in further view of Bhide</u>

The Office at 8 states:

> **8. Claim 4 is rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 1 above, and further in view of Bhide et al U.S. Patent No. 5,852,717.**
>
> As to claim 4, Vellanki et al does not teach initiating a HTTP connection that comprises initiating a HTTP connection to a predefined address using port 80.
>
> Bhide et al teaches initiating a HTTP connection that comprises initiating a HTTP connection to a predefined address using port 80 [column 5, lines 9-21].
>
> Therefore, it would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Vellanki et al so that if a HTTP connection were to initiate between a client and server, it would have used a predefined address using port 80.
>
> It would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Vellanki et al by the teaching of Bhide et al because it is well known in the art that a HTTP connection uses port 80. Establishing a connection involves one round-trip time from the client to the server as the client requests to open a network connection and the server responds that a network connection has been opened [column 5, lines 9-21].

(Emphasis in original.)

The cited reference states in part:

[column 5, lines 9-21] - BHIDE

The Web utilizes the HTTP client/server protocol, which is a request-response protocol. HTTP transactions include four stages: connection, request, response, and disconnection. In the connection stage, the client attempts to open a network connection to the server. Unless otherwise specified, HTTP attempts to use port 80 on the server for this connection. Establishing a connection involves one round-trip time from the client to the server as the client requests to open a network connection and the server responds that a network connection has been opened. Although the discussion herein focuses on version 1.0 of HTTP, the invention is not limited to any version of HTTP or to HTTP specifically.

Applicant's claim 4 recites:

4. The method according to claim 2, wherein initiating a HTTP connection comprises initiating a HTTP connection to a predefined address using port 80.

Applicant submits that …

Claim 4 is dependent on claim 2, which is dependent on claim 1. The issue of a 102(b) Vellanki rejection for claim 1 is addressed above and incorporated herein. As discussed above, Applicant's independent claim 1 recites a limitation wherein the second connection is different than the first connection and the third connection is different than the second connection and the first connection upon which dependent claim 4 depends. Vellanki fails to disclose a second and third connections that are each different. <u>Bhide et al ("Bhide") also fails to disclose a second and third connections that are each different. Vellanki in view of Bhide also fails to disclose Applicant's limitation wherein the second connection is different than the first connection and the third connection is different than the second connection and the first connection.</u> Applicant respectfully requests allowance of claim 4.

Claims 5-9 Rejection under 35 U.S.C. § 103(a)

The Office at 9 states:

9. Claims 5-9 are rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 1 above, and further in view of Fuh et al U.S. Patent No. 6,609,154 B1.

(Emphasis in original.)

Claims 5-7, and 9 Rejection under 35 U.S.C. § 103(a) – Vellanki in further view of Fuh

The Office at 9 states: (continued)

As to claims 5-7 and 9, Vellanki et al does not teach that initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address and port. Vellanki et al does not teach that determining a likely proxy address and port further comprises packet sniffing. Vellanki et al does not teach that packet sniffing further comprises: sampling packets; extracting information from the sampled packets; and building a database of likely proxy addresses and ports. Vellanki et al does not teach that extracting information from the sampled packets comprises examining TCP packets for HTTP data.

Fuh et al teaches initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address and port [column 13, lines 3-14]. Fuh et al teaches that determining a likely proxy address and port further comprises packet sniffing [column 9, lines 51-67]. Fuh et al teaches that packet sniffing further comprises: sampling packets; extracting information from the sampled packets; and building a database of likely proxy addresses and ports [column 9, lines 51-67]. Fuh et al teaches that extracting information from the sampled packets comprises examining TCP packets for HTTP data [column 9, lines 51-67].

Therefore, it would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Vellanki et al so that there would have been a HTTP connection initiated via a proxy connection that would have determined a likely proxy address and port. Packet sniffing would have occurred during the determining step of the proxy address and port. The firewall packet sniffing would have included sampling packets, extracting information from the packets and building a database of likely proxy addresses and ports. The extracted information would have come from examining TCP packets for HTTP data.

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Vellanki et al by the teaching of Fuh et al because it makes sure that the client is authorized to communicate with a network resource [column 3, lines 54- 60].

The cited reference states in part:

[column 13, lines 3-14] - Fuh

As shown in block 738, the process waits. For example, after Authentication Proxy 400 sends the "Authentication Success" message 524 to User 302, the Authentication Proxy enters a wait state for a short, pre-determined period of time. During the wait state, a short period of time is allowed to elapse to enable client 306 and firewall router 210 to communicate handshaking messages and carry out related processing associated with establishing an HTTP connection. The delay period also allows the firewall router enough time to execute any commands that are issued as part of block 734. In one embodiment, a period of three (3) seconds elapses.

[column 9, lines 51-67] - Fuh

Access control lists filter packets and can prevent certain packets from entering or exiting a network. Each ACL is a list of information that firewall router 210 may use to determine whether packets arriving at or sent from a particular interface may be communicated within or outside the firewall router. For example, in an embodiment, input ACL 424 may comprise a list of IP addresses and types of allowable client protocols. Assume that firewall router 210 receives an inbound packet from client 306 at external interface 420 that is intended for target server 222. If the IP address of client 306 is not stored in input ACL 424, then firewall router 210 will not forward the packet further within the circuitry or software of the firewall router. Output ACL 426 similarly controls the delivery of packets from firewall router 210 to resources located outside external interface 420. Input ACL 428 and output ACL 430 govern packet flow to or from internal interface 422.

[column 3, lines 54-60] - Fuh

In another feature, determining whether the client is authorized to communicate with the network resource comprises the steps of: determining whether a source IP address of the client in the request matches information in a filtering mechanism of the network device; and if so, determining whether the source IP address matches the authorization information stored in the network device.

Applicant's claims 5-7 and 9 recite:

5. The method according to claim 2, wherein initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address and port.

6. The method according to claim 5, wherein determining a likely proxy address and port further comprises packet sniffing.

7. The method according to claim 6, wherein packet sniffing further comprises:

sampling packets;

extracting information from the sampled packets; and

building a database of likely proxy addresses and ports.

9. The method according to claim 7, wherein extracting information from the sampled packets comprises examining TCP packets for HTTP data.

Firstly, claims 5-7, and 9 are dependent on claim 2, which is dependent on claim 1. The issue of a 102(b) Vellanki rejection for claim 1 is addressed above and incorporated herein. As discussed above, Applicant's independent claim 1 recites a limitation wherein the second connection is different than the first connection and the third connection is different than the second connection and the first connection upon which dependent claims 5-7, and 9 depend. Vellanki fails to disclose a second and third connections that are each different. Fuh et al ("Fuh") also fails to disclose a second and third connections that are each different. Vellanki in view of Fuh also fails to disclose Applicant's limitation wherein the second connection is different than the first connection and the third connection is different than the second connection and the first connection. Applicant respectfully requests allowance of claims 5-7, and 9.

Specifically with respect to claim 5

Applicant's claim 5 recites:

5. The method according to claim 2, wherein initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address and port.
[Emphasis added.]

Applicant's claim 5 is dependent on claim 2, which is in turn dependent on claim 1. The issue of a 102(b) Vellanki rejection for claims 1, and 2 are addressed above and incorporated herein.

Applicant submits that Fuh is fundamentally different than Applicant's claim 5. While Applicant teaches determining a likely proxy address and port, Fuh (see Abstract) on the other hand teaches "network access control" and "intercept network traffic." Further, Fuh Figure 2 at 210 clearly shows intercepting traffic to/from 206 and 216 and the specification details authentication based on AA server 218 and Database 220. Network access control and intercepting network traffic (Fuh) is not the same as determining a likely proxy address and port (Applicant's claim 5).

Additionally, while the Office states "Fuh et al teaches initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address and port [column 13, lines 3-14]." Applicant submits that the cited lines discuss the authentication process and do not teach anything about determining a likely proxy address and port as in Applicant's claim 5.

Finally, modifying Vellanki with Fuh does not disclose or make obvious the "firewall" aspect of claim 1 or the "determining a likely proxy address and port" aspect of claim 5. Applicant respectfully requests removal of this rejection for claim 5 and claims 6-9 which are dependent on claim 5.

Specifically with respect to claim 6

Applicant's claim 6 recites:

6. The method according to claim 5, wherein determining a likely proxy address and port further comprises packet sniffing.

[Emphasis added.]

The Office cites Fuh for "Fuh et al teaches that determining a likely proxy address and port further comprises packet sniffing [column 9, lines 51-67]."

Applicant submits that Fuh actually teaches away from Applicant's claim 6. While Applicant teaches sniffing packets which does not involve altering in any way the communication, Fuh (see Abstract) on the other hand teaches network access control and intercepting network traffic. Intercepting network traffic (Fuh) is the *antithesis* of packet sniffing (Applicant).

Further, Applicant submits that the cited lines discuss the Authentication and Authorization process and do not teach anything about determining a likely proxy address and port further comprises packet sniffing as in Applicant's claim 6. Fuh at the lines cited specifically says "Access control lists filter packets .... If the IP address of client 306 is not stored in input ACL 424, then firewall router 210 will not forward the packet further within the circuitry or software of the firewall router. Output ACL 426 similarly controls the delivery of packets from firewall router 210 to resources located outside external interface 420."

[Emphases added.]

Fuh teaches away from packet sniffing and deals with filtering packets, not forwarding packets, and controlling delivery.

Finally, modifying Vellanki with Fuh does not disclose or make obvious the "packet sniffing" aspect of claim 6. Applicant respectfully requests removal of this rejection for claim 6 and claims 7-9 which are dependent on claim 6.

Specifically with respect to claim 7

Applicant's claim 7 recites:

7. The method according to claim 6, wherein packet sniffing further comprises:

sampling packets;

extracting information from the sampled packets; and

building a database of likely proxy addresses and ports.

The Office cites Fuh for "Fuh et al teaches that packet sniffing further comprises: sampling packets; extracting information from the sampled packets; and building a database of likely proxy addresses and ports [column 9, lines 51-67]."

Applicant submits that the cited lines discuss the Authentication and Authorization process and do not teach anything about sampling packets; extracting information from the sampled packets; and building a database of likely proxy addresses and ports as in Applicant's claim 7.

Fuh at the lines cited specifically says "Access control lists filter packets .... If the IP address of client 306 is not stored in input ACL 424, then firewall router 210 will not forward the packet further within the circuitry or software of the firewall router. Output ACL 426 <u>similarly controls the delivery of packets from firewall router</u> 210 to resources located outside external interface 420."

[Emphases added.]

As discussed above for claim 6, <u>Fuh teaches away</u> from packet sniffing and deals with filtering packets, not forwarding packets, and controlling delivery.

Finally, modifying Vellanki with Fuh does not disclose or make obvious the "sampling packets; extracting information from the sampled packets; and building a database of likely proxy addresses and ports" aspect of claim 7. Applicant respectfully requests removal of this rejection for claim 7 and claims 8-9 which are dependent on claim 7.

<u>Claim 8 Rejection under 35 U.S.C. § 103(a) – Vellanki in further view of Fuh</u>

The Office at 9 states: (continued)

As to claim 8, Vellanki et al teaches that extracting information from the sampled packets comprises extracting TCP port information [column 1 line 50 to column 2 line 3].

The cited reference states in part:

[column 1 line 50 to column 2 line 3] - Vellanki

Several authentication and authorization mechanisms are suitable for use with operating systems that are used by network devices, such as the Internetworking Operating System ("IOS") commercially available from Cisco Systems, Inc. However, most prior authentication and authorization mechanisms are associated with dial-up interfaces, which can create network security problems. In a dial-up configuration, a remote client uses a telephone line and modem to dial up a compatible modem that is coupled to a server of the network that the remote client wishes to access. In another dial-up configuration, a remote client first establishes a dial-up connection to a server associated with an Internet Service Provider, and that server then connects to the network server through the global, public, packet-switched internetwork known as the Internet. In this configuration, the network server is coupled directly or indirectly to the Internet.

Unfortunately, information requests and other traffic directed at a network server from the Internet is normally considered risky, untrusted traffic. An organization that owns or operates a network server can protect itself from unauthorized users or from unwanted traffic from the Internet by using a firewall. . .

Applicant's claim 8 recites:

8. The method according to claim 7, wherein extracting information from the sampled packets comprises extracting TCP port information.

Claim 8 is dependent on claim 7, which is dependent on claim 6, which is dependent on claim 5, which is dependent on claim 2, which is dependent on claim 1. The issue of a 102(b) Vellanki rejection for claim 1 is addressed above and incorporated herein.

Applicant submits that nowhere in the cited section does Fuh mention what Applicant has claimed. Furthermore, nowhere does Fuh mention "extracting" TCP port information from sampled packets. Finally, modifying Vellanki with Fuh does not disclose or make obvious "wherein extracting information from the sampled packets comprises extracting TCP port information" limitation of claim 8. Applicant respectfully requests removal of this rejection for claim 8, and allowance of claim 8.

Claims 11 and 13 Rejection under 35 U.S.C. § 103(a) – Vellanki in further view of Fuh

The Office at 10 states:

10. Claims 11 and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 1 above, and further in view of Fuh et al U.S. Patent No. 6,609,154 B1.

(Emphasis in original.)

As to claims 11 and 13, Vellanki et al does not teach that initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address by sampling packets and extracting IP and Ethernet addresses.

Fuh et al teaches initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address by sampling packets and extracting IP and Ethernet addresses [column 9, lines 51-67].

Therefore, it would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Vellanki et al so that a HTTP connection would have been initiated via a proxy connection and proxy addresses would have been determined by sampling packets and extracting IP and Ethernet address.

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Vellanki et al by the teaching of Fuh et al because it makes sure that the client is authorized to communicate with a network resource [column 3, lines 54- 60].

The cited reference states in part:

[column 9, lines 51-67] - Fuh

Access control lists filter packets and can prevent certain packets from entering or exiting a network. Each ACL is a list of information that firewall router 210 may use to determine whether packets arriving at or sent from a particular interface may be communicated within or outside the firewall router. For example, in an embodiment, input ACL 424 may comprise a list of IP addresses and types of allowable client protocols. Assume that firewall router 210 receives an inbound packet from client 306 at external interface 420 that is intended for target server 222. If the IP address of client 306 is not stored in input ACL 424, then firewall router 210 will not forward the packet further within the circuitry or software of the firewall router. Output ACL 426 similarly controls the delivery of packets from firewall router 210 to resources located outside external interface 420. Input ACL 428 and output ACL 430 govern packet flow to or from internal interface 422.

[column 3, lines 54-60] - Fuh

In another feature, determining whether the client is authorized to communicate with the network resource comprises the steps of: determining whether a source IP address of the client in the request matches information in a filtering mechanism of the network device; and if so, determining whether the source IP address matches the authorization information stored in the network device.

Specifically with respect to claim 11

Applicant's claims 11 recites:

11. The method according to claim 10, wherein initiating a HTTP connection via
a proxy connection further comprises determining a likely proxy address by sampling packets and extracting IP addresses.

As detailed above, Applicant submits that the cited lines discuss the Authentication and Authorization process and do not teach anything about determining a likely proxy address by sampling packets and extracting IP addresses as in Applicant's claim 11. Fuh at the lines cited specifically says "Access control lists filter packets .... If the IP address of client 306 is not stored in input ACL 424, then firewall router 210 will not forward the packet further within the circuitry or software of the firewall router. Output ACL 426 similarly controls the delivery of packets from firewall router 210 to resources located outside external interface 420."

[Emphases added.]

Filtering packets and not forwarding packets (Fuh) is not the same as sampling packets or extracting IP addresses (as in Applicant's claim 11).

Finally, modifying Vellanki with Fuh does not disclose or make obvious the "determining a likely proxy address by sampling packets and extracting IP addresses" aspect of claim 11. Applicant respectfully requests removal of this rejection for claim 11.

Specifically with respect to claim 13

Applicant's claim 13 recites:

13. The method according to claim 12, wherein initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address by sampling packets and extracting Ethernet addresses.

As detailed above, Applicant submits that the cited lines discuss the Authentication and Authorization process and do not teach anything about determining a likely proxy address by sampling packets and extracting Ethernet addresses as in Applicant's claim 13. Fuh at the lines cited specifically says "Access control lists filter packets .... If the IP address of client 306 is not stored in input ACL 424, then firewall router 210 will not forward the packet further within the circuitry or software of the firewall router. Output ACL 426 similarly controls the delivery of packets from firewall router 210 to resources located outside external interface 420."

[Emphases added.]

Filtering packets and not forwarding packets (Fuh) is not the same as sampling packets or extracting Ethernet addresses (as in Applicant's claim 13).

Finally, modifying Vellanki with Fuh does not disclose or make obvious the "determining a likely proxy address by sampling packets and extracting Ethernet addresses" aspect of claim 13. Applicant respectfully requests removal of this rejection for claim 13.

Claim 16 Rejection under 35 U.S.C. § 103(a) – Vellanki in view of Linden

The Office at 11 states:

11. Claim 16 is rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 14 above, and further in view of Linden et al U.S. Patent No. 6,549,773 B1.

(Emphasis in original.)

As to claim 16, Vellanki et al teaches examining network traffic [column 5, lines 47-67].

Vellanki et al does not teach building a database of parameters likely to allow establishment of a HTTP connection via a proxy connection.

Linden et al teaches building a database of parameters likely to allow establishment of a HTTP connection via a proxy connection [column 5, lines 16-26].

Therefore, it would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Vellanki et al so that a database would have been built of parameters likely to allow establishment of a HTTP connection via a proxy connection.

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Vellanki et al by the teaching of Linden et al because it is possible to efficiently utilize functions connected with the HTTP data transmission protocol of the WSP/B protocol already known as such. These include, for example, GET, PUT, and POST requests. Consequently, the header fields of the HTTP protocol can also be utilized in the data transmission, as well as the headers of the HTTP protocol for authentication. Correspondingly, it is possible to utilize efficiently the methods of the WWW communication network for authorization or data transmission [column 5, lines 16-26].

The cited references state in part:

[column 5, lines 47-67] - Vellanki

The present invention will now be described in detail with reference to a few preferred embodiments thereof as illustrated in the accompanying drawings. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without some or all of these specific details. In other instances, well known process steps have not been described in detail in order to not unnecessarily obscure the present invention.

In accordance with one aspect of the present invention, the client computer in a heterogeneous client-server computer network (e.g., client computer 106 in FIG. 1) is provided with an autodetect mechanism. When executed, the autodetect mechanism advantageously permits client computer 106 to select, in an efficient and automatic manner, the most advantageous protocol for communication between the client computer and its server. Once the most advantageous protocol is selected, parameters pertaining to the selected protocol are saved to enable the client computer, in future sessions, to employ the same selected protocol for communication.

[column 5, lines 16-26]- LINDEN

A particular advantage of the invention e.g. in connection with the WAP application protocol is that it is possible to efficiently utilize functions connected with the HTTP data transmission protocol of the WSP/B protocol already known as such. These include, for example, GET, PUT, and POST requests. Consequently, the header fields of the HTTP protocol can also be utilized in the data transmission, as well as the headers of the HTTP protocol for authentication. Correspondingly, it is possible to utilize efficiently the methods of the WWW communication network for authorization or data transmission.

Applicant submits that <u>the cited lines of Linden</u> et al ("Linden") discuss protocols and <u>do not teach anything about building a database of parameters likely to allow establishment of a HTTP connection via a proxy connection as in Applicant's claim 16</u>.

Applicant submits that Vellanki in view of Linden does not make obvious what is in Applicant's claim 16. Applicant therefore respectfully requests removal of this rejection for claim 16.

Claims 24 and 25 under 35 U.S.C. § 103(a) – Vellanki further in view of Fuh

The Office at 12 states:

12. Claims 24 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 22 above, and further in view of Fuh et al U.S. Patent No. 6,609,154 B1.

(Emphasis in original.)

As to claims 24 and 25, Vellanki et al does not teach means for initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address by sniffing packets and extracting information from the packets. Vellanki et al does not teach means for initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address by receiving information from a computer connected to the firewall.

Fuh teaches means for initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address by sniffing packets and extracting information from the packets [column 9, lines 51-67]. Fuh teaches means for initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address by receiving information from a computer connected to the firewall [column 9, lines 51-67].

Therefore, it would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Vellanki et al so that a HTTP connection would have been initiated via a proxy connection. The firewall would have sniffed packets and extracted information from the packets. Proxy addresses would have been determined by receiving information from the computer connected to the firewall.

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Vellanki et al by the teaching of Fuh et al because it makes sure that the client is authorized to communicate with a network resource [column 3, lines 54-60].

The cited reference states in part:

[column 9, lines 51-67] - Fuh

Access control lists filter packets and can prevent certain packets from entering or exiting a network. Each ACL is a list of information that firewall router 210 may use to determine whether packets arriving at or sent from a particular interface may be communicated within or outside the firewall router. For example, in an embodiment, input ACL 424 may comprise a list of IP addresses and types of allowable client protocols. Assume that firewall router 210 receives an inbound packet from client 306 at external interface 420 that is intended for target server 222. If the IP address of client 306 is not stored in input ACL 424, then firewall router 210 will not forward the packet further within the circuitry or software of the firewall router. Output ACL 426 similarly controls the delivery of packets from firewall router 210 to resources located outside external interface 420. Input ACL 428 and output ACL 430 govern packet flow to or from internal interface 422.

[column 3, lines 54-60] - Fuh

In another feature, determining whether the client is authorized to communicate with the network resource comprises the steps of: determining whether a source IP address of the client in the request matches information in a filtering mechanism of the network device; and if so, determining whether the source IP address matches the authorization information stored in the network device.

Specifically with respect to claim 24

Applicant's claim 24 recites:

24. The apparatus of claim 23, wherein means for initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address by sniffing packets and extracting information from the packets.

[Emphasis added.]

Applicant submits that Fuh actually teaches away from Applicant's claim 24. While Applicant teaches sniffing packets which does not involve altering in any way the communication, Fuh (see Abstract) on the other hand teaches network access control and intercepting network traffic. Intercepting network traffic (Fuh) is the *antithesis* of packet sniffing (Applicant).

Further, Applicant submits that the cited lines discuss the Authentication and Authorization process and do not teach anything about determining a likely proxy address by sniffing packets and extracting information from the packets as in Applicant's claim 24.

Fuh at the lines cited specifically says "Access control lists filter packets .... If the IP address of client 306 is not stored in input ACL 424, then firewall router 210 will not forward the packet further within the circuitry or software of the firewall router. Output ACL 426 similarly controls the delivery of packets from firewall router 210 to resources located outside external interface 420."

[Emphases added.]

Fuh teaches away from packet sniffing and deals with filtering packets, not forwarding packets, and controlling delivery.

Claim 24 is dependent on claim 23, which is dependent on claim 22. The issue of a 102(b) Vellanki rejection for claim 22 is addressed above and incorporated herein. As discussed above Applicant's independent claim 22 recites a limitation wherein the second connection is different than the first connection and the third connection is different than the second connection and the first connection upon which dependent claim 24 depends. Fuh fails to disclose a second and third connections that are each different. Fuh also fails to disclose a second and third connections that are each different. Vellanki in view of Fuh also fails to disclose Applicant's limitation wherein the second connection is different than the first connection and the third connection is different than the second connection and the first connection. Applicant respectfully requests allowance of claim 24.

Finally, modifying Vellanki with Fuh does not disclose or make obvious the "packet sniffing" aspect of claim 24. Applicant respectfully requests removal of this rejection for claim 24.

Specifically with respect to claim 25

Applicant's claim 25 recites:

25. The apparatus of claim 23, wherein means for initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address by receiving information from a computer connected to the firewall.

[Emphasis added.]

The Office (page 9, paragraph 11) states:

Fuh teaches means for initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address by receiving information from a computer connected to the firewall [column 9, lines 51-67].

Applicant submits that <u>the cited lines</u> discuss the Authentication and Authorization process and <u>do not teach anything about</u> determining a likely proxy address by <u>receiving information from a computer connected to the firewall</u> as in Applicant's claim 25.

Fuh at the lines cited specifically says "Access control lists <u>filter packets</u> .... If the IP address of client 306 is not stored in input ACL 424, then firewall <u>router 210 will not forward the packet</u> further within the circuitry or software of the firewall router. Output ACL 426 <u>similarly controls the delivery of packets from firewall router</u> 210 to resources located outside external interface 420."

[Emphases added.]

Claim 25 is dependent on claim 23, which is dependent on claim 22. The issue of a 102(b) Vellanki rejection for claim 22 is addressed above and incorporated herein. As discussed above, Applicant's independent claim 22 recites a limitation wherein the second connection is different than the first connection and the third connection is different than the second connection and the first connection upon which dependent claim 25 depends. Fuh fails to disclose a first and second connections that are each different. Fuh also fails to disclose a second and third connections that are each different. Vellanki in view of Fuh also fails to disclose Applicant's limitation wherein the second connection is different than the first connection and the third connection is different than the second connection and the first connection. Applicant respectfully requests allowance of claim 25.

Finally, modifying Vellanki with Fuh does not disclose or make obvious the "receiving information from a computer connected to the firewall" aspect of claim 25. Applicant respectfully requests removal of this rejection for claim 25.

Claim 26 under 35 U.S.C. § 103(a) – Vellanki in view of Montenegro

The Office at 13 states:

13. Claim 26 is rejected under 35 U.S.C. 103(a) as being unpatentable over Vellanki et al U.S. Patent No. 5,999,979 as applied to claim 22 above, and further in view of Montenegro U.S. Patent No. 6,233,688 B1. (Emphasis in original.)

As to claim 26, Vellanki et al does not teach means for updating firewall traversal strategies.

Montenegro teaches means for updating firewall traversal strategies [column 6, lines 49-65].

Therefore, it would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Vellanki et al so that there would have been a firewall that had means for updated firewall traversal strategies.

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Vellanki et al by the teaching of Montenegro because it keeps the firewall up to date as far as addressed to block so that the client is not compromised at any time [column 2, lines 7-21].

The cited reference states:

[column 6, lines 49-65] - Montenegro

The first step in creating a transparency between the firewall traversal for remote access and the client application is to obtain the appropriate RAFT URL (step 510). The discovery of the specific RAFT URL to use is not a subject of the essential invention. Obtaining a RAFT URL may be achieved in several ways: (1) obtaining it in person from a system administrator, (2) visiting a special web page where an authenticated user may retrieve the appropriate RAFT URL from the firewall, (3) querying a directory service such as LDAP (Lightweight Directory Access Protocol) or (4) may be preconfigured into the client application or system. The appropriate RAFT URL will designate parameters allowing the client system to get private intranet resources through its data transport mechanisms (IP stack, sockets, etc.).

(Emphases added.)

[column 2, lines 7-21] - Montenegro

The invention provides a generic naming scheme for remote access and firewall traversal in the form of a uniform resource locator (RAFT URL). The RAFT URL may be provided to any client application, regardless of compatibility with the remote access/firewall traversal method, which then launches another executable module. The executable module performs the remote access/firewall traversal method and interacts with the operating environment to obtain data transport mechanisms. These mechanisms permit the client application to transact with private resources beyond the firewall. The remote access/firewall traversal procedure is made transparent to the client application, and thus, a wider array of client applications may be chosen for the data session with the resources beyond the firewall.

(Emphases added.)

Applicant's claim 26 recites:

26. The apparatus of claim 22, further comprising means for updating firewall traversal strategies.

Applicant submits that Montenegro does not teach "means for updating firewall traversal strategies". To the contrary, at the cited reference lines Montenegro assumes that a RAFT URL exists (The RAFT URL may be provided to any client application) and then uses this to launch an executable to get through a firewall (The executable module performs the remote access/firewall traversal method). Montenegro does not disclose a means for "updating firewall traversal strategies" as Applicant has claimed.

Claim 26 is dependent on claim 22. The issue of a 102(b) Vellanki rejection for claim 22 is addressed above and incorporated herein. As discussed above Applicant's independent claim 22 recites a limitation wherein the second connection is different than the first connection and the third connection is different than the second connection and the first connection upon which dependent claim 26 depends. Montenegro fails to disclose a first and second connections that are each different. Montenegro also fails to disclose a second and third connections that are each different. Vellanki in view of Montenegro also fails to disclose Applicant's limitation wherein the second connection is different than the first connection and the third connection is different than the second connection and the first connection. Applicant respectfully requests allowance of claim 26.

Finally, modifying Vellanki with Montenegro does not disclose or make obvious "means for updating firewall traversal strategies" aspect of claim 26. Applicant respectfully requests removal of this rejection for claim 26.

CONCLUSION

Applicant submits that the rejection of dependent claims not specifically addressed, are addressed by Applicant's arguments to the independent claims on which they depend.

Applicant respectfully submits that the appealed claims in this application are patentable, and requests that the Board of Patent Appeals and Interferences direct allowance of all claims.

Respectfully submitted,

Heimlich Law

08/15/2007 /Alan Heimlich/

Date Alan Heimlich / Reg 48808

Attorney for Applicant(s)

Customer No. 40418

5952 Dial Way
San Jose, CA 95129

Tel: 408 253-3860
Eml: alanheimlich@heimlichlaw.com

VIII. CLAIMS APPENDIX

The claims involved in this appeal (all pending claims) are as follows:

CLAIMS

What is claimed is:

1. A method for traversing a firewall, comprising:

initiating a first connection to go through said firewall;

evaluating the first connection for a response from a remote system indicating a successful first connection;

initiating a second connection to go through said firewall if a successful first connection is not established, wherein said second connection is different than said first connection;

evaluating the second connection for a response from a remote system indicating a successful second connection;

initiating a third connection to go through said firewall if a successful second connection is not established, wherein said third connection is different than said second connection and said first connection; and

evaluating the third connection for a response from a remote system indicating a successful third connection.

2. The method of claim 1, wherein the first connection, the second connection, and the third connection is selected from the group consisting of Transmission Control Protocol (TCP) connection, User Datagram Protocol (UDP) connection, hypertext transfer protocol (HTTP)

connection, hypertext transfer protocol (HTTP) connection via a proxy connection, and Internet Control Message Protocol (ICMP) connection.

3. The method according to claim 2, wherein initiating a TCP connection comprises initiating a TCP connection to a predefined address and port.

4. The method according to claim 2, wherein initiating a HTTP connection comprises initiating a HTTP connection to a predefined address using port 80.

5. The method according to claim 2, wherein initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address and port.

6. The method according to claim 5, wherein determining a likely proxy address and port further comprises packet sniffing.

7. The method according to claim 6, wherein packet sniffing further comprises:

sampling packets;

extracting information from the sampled packets; and

building a database of likely proxy addresses and ports.

8. The method according to claim 7, wherein extracting information from the sampled packets comprises extracting TCP port information.

9. The method according to claim 7, wherein extracting information from the sampled packets comprises examining TCP packets for HTTP data.

10. The method of claim 2 further comprising using Internet Protocol (IP).

11. The method according to claim 10, wherein initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address by sampling packets and extracting IP addresses.

12. The method of claim 2 further comprising using Ethernet with the Transmission Control Protocol (TCP).

13. The method according to claim 12, wherein initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address by sampling packets and extracting Ethernet addresses.

14. A machine-readable medium having stored thereon instructions, which when executed by a processor, causes said processor to perform the following:

initiate a first connection to go through a firewall;

evaluate the first connection for a response from a remote system indicating a successful first connection;

initiate a second connection to go through said firewall if a successful first connection is not established, wherein said second connection is different than said first connection;

evaluate the second connection for a response from a remote system indicating a successful second connection;

initiate a third connection to go through said firewall if a successful second connection is not established, wherein said third connection is different than said second connection and said first connection; and

evaluate the third connection for a response from a remote system indicating a successful third connection.

15. The machine-readable medium according to claim 14, further configuring said processor to perform the following:

implement the first connection, the second connection, and the third connection selected from the group consisting of Transmission Control Protocol (TCP) connection, User Datagram Protocol (UDP) connection, hypertext transfer protocol (HTTP) connection, hypertext transfer protocol (HTTP) proxy connection, and Internet Control Message Protocol (ICMP) connection.

16. The machine-readable medium according to claim 15, further configuring said processor to perform the following:

examine network traffic; and

build a database of parameters likely to allow establishment of a HTTP connection via a proxy connection.

17. A firewall traversal system comprising:

a main system coupled to storage;

a communication subsystem coupled to the main system and a communication medium on one side of a firewall;

a packet examining subsystem coupled to the communication subsystem; and

a database system coupled to the packet examining subsystem and the main system.

18. The system of claim 17, wherein the packet examining subsystem extracts port information.

19. The system of claim 18, wherein the packet examining subsystem extracts the port information based upon examining packet data content.

20. The system of claim 17, wherein the packet examining subsystem extracts address information.

21. The system of claim 20, wherein the packet examining subsystem extracts the address information based upon examining packet data content.

22. A method for traversing a firewall, comprising:

means for initiating a first connection to go through said firewall;

means for evaluating the first connection for a response from a remote system indicating a successful first connection;

means for initiating a second connection to go through said firewall if a successful first connection is not established, wherein said second connection is different than said first connection;

means for evaluating the second connection for a response from a remote system indicating a successful second connection;

means for initiating a third connection to go through said firewall if a successful second connection is not established, wherein said third connection is different than said second connection and said first connection; and

means for evaluating the third connection for a response from a remote system indicating a successful third connection.

23. The apparatus of claim 22, wherein means for initiating the first connection, means for initiating the second connection, and means for initiating the third connection further comprises means for initiating a connection selected from the group consisting of Transmission Control Protocol (TCP) connection, User Datagram Protocol (UDP) connection, hypertext transfer protocol (HTTP) connection, hypertext transfer protocol (HTTP) proxy connection, and Internet Control Message Protocol (ICMP) connection.

24. The apparatus of claim 23, wherein means for initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address by sniffing packets and extracting information from the packets.

25. The apparatus of claim 23, wherein means for initiating a HTTP connection via a proxy connection further comprises determining a likely proxy address by receiving information from a computer connected to the firewall.

26. The apparatus of claim 22, further comprising means for updating firewall traversal strategies.

IX. EVIDENCE APPENDIX

Grouping of any Evidence for Claim purposes is for the convenience of reduced duplication and is NOT to be interpreted as the Grouping of Claims for Arguments under 37 C.F.R. § 41.37.

(A) Evidence for Claims 1-26 – Relied Upon

The following item (1) listed below is hereby entered as evidence relied upon by the Examiner as to grounds of rejection for claims 1-26, to be reviewed on appeal. Also listed for each item is where said evidence was entered into the record by the Examiner.

(1) Copy of US Patent Number 5,999,979 ("Vellanki et al"). This evidence was entered into the record by the Examiner on page 3 at 7 of the Office Action mailed 08/10/2006.

Copies of all References follows.

//

US005999979A

United States Patent [19]

Vellanki et al.

[11] **Patent Number: 5,999,979**

[45] **Date of Patent: Dec. 7, 1999**

[54] **METHOD AND APPARATUS FOR DETERMINING A MOST ADVANTAGEOUS PROTOCOL FOR USE IN A COMPUTER NETWORK**

[75] Inventors: **Srinivas Prasad Vellanki**, Milpitas; **Anthony William Cannon**, Mountain View; **Hemanth Srinivas Ravi**, Milpitas; **Anders Edgar Klemets**, Sunnyvale, all of Calif.

[73] Assignee: **Microsoft Corporation**, Redmond, Wash.

[21] Appl. No.: **08/818,769**

[22] Filed: **Mar. 14, 1997**

Related U.S. Application Data

[60] Provisional application No. 60/036,661, Jan. 30, 1997, and provisional application No. 60/036,662, Jan. 30, 1997.

[51] **Int. Cl.**^{6} **G06F 15/16**

[52] **U.S. Cl.** **709/232**; 709/237

[58] **Field of Search** 395/200.57, 200.58, 395/200.6, 200.76, 200.62; 710/11; 709/228, 227, 230, 231, 232, 237

[56] **References Cited**

U.S. PATENT DOCUMENTS

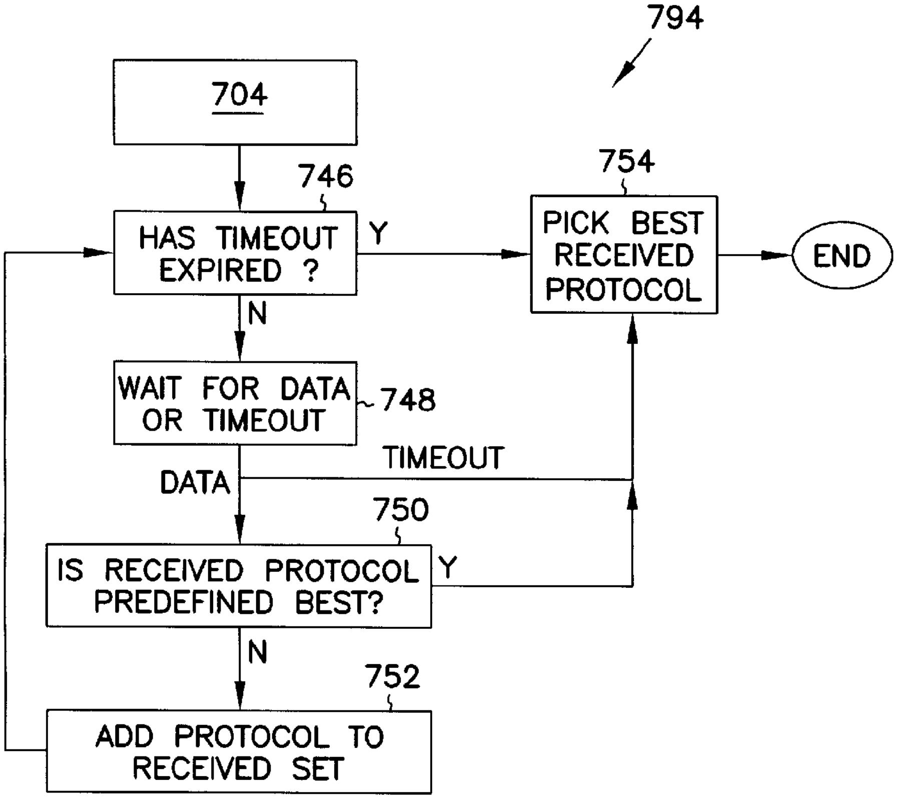
4,931,250	6/1990	Greszczuk	375/8
5,202,899	4/1993	Walsh	375/8
5,349,649	9/1994	Iijima	395/275
5,557,724	9/1996	Sampat et al.	395/157
5,687,174	11/1997	Edem et al.	370/446

Primary Examiner—Stuart S. Levy
Assistant Examiner—Kenneth W. Fields
Attorney, Agent, or Firm—Schwegman, Lundberg, Woessner & Kluth P.A.

[57] **ABSTRACT**

A method in a computer network for automatically detecting a most advantageous protocol for communication by a client computer, said client computer being configured to be coupled to a server computer via a computer network. The method includes initiating a plurality of protocol threads for sending from the client computer to the server computer, a plurality of data requests. Each of the data requests employs a different protocol and a different connection. The data requests are configured to solicit, responsive to the data requests, a set of responses from the server computer. Each of the responses employs a protocol associated with a respective one of the data requests. The method further includes receiving at the client computer at least a subset of the responses. The method also includes initiating a control thread at the client computer. The control thread monitors the subset of the responses as each response is received from the server computer to select the most advantageous protocol from protocols associated with the subset of the responses, wherein the most advantageous protocol is determined based on a predefined protocol priority.

23 Claims, 11 Drawing Sheets


794
704
746
HAS TIMEOUT EXPIRED ?
Y
754
PICK BEST RECEIVED PROTOCOL
END
N
WAIT FOR DATA OR TIMEOUT
748
TIMEOUT
DATA
750
IS RECEIVED PROTOCOL PREDEFINED BEST?
Y
N
752
ADD PROTOCOL TO RECEIVED SET

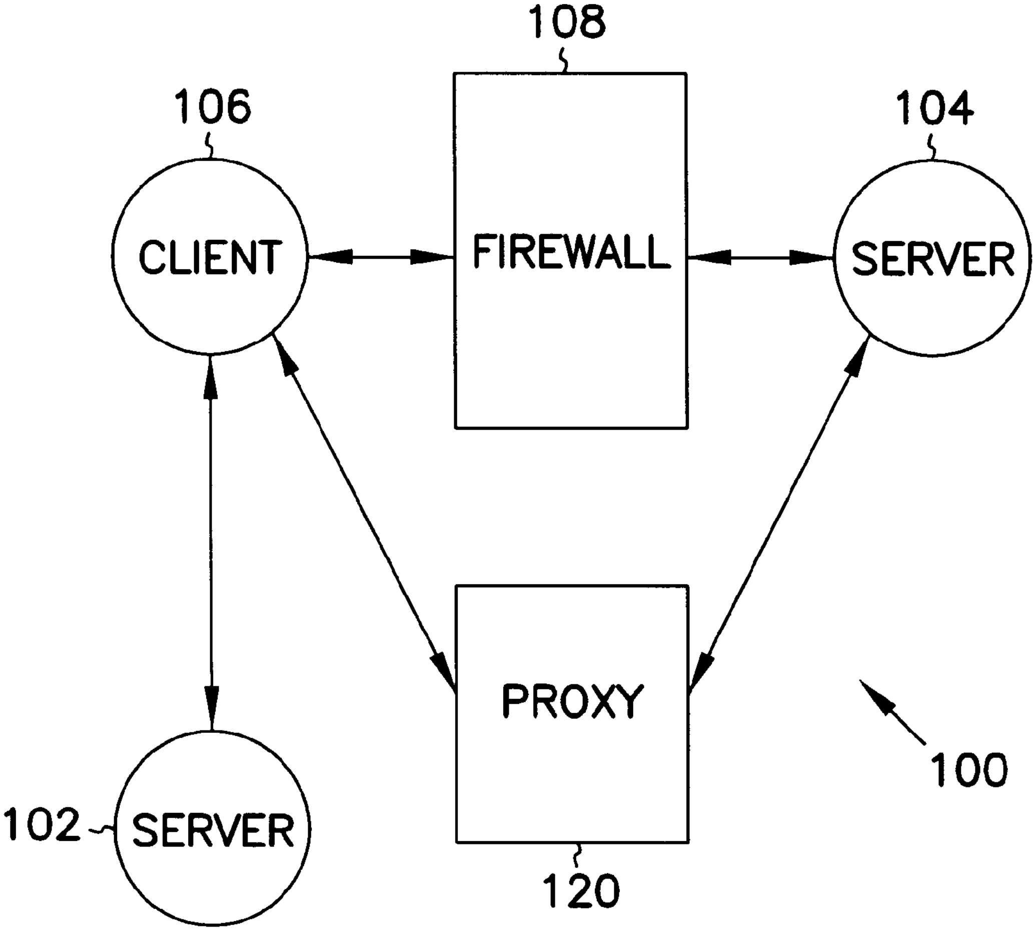


FIG. 1

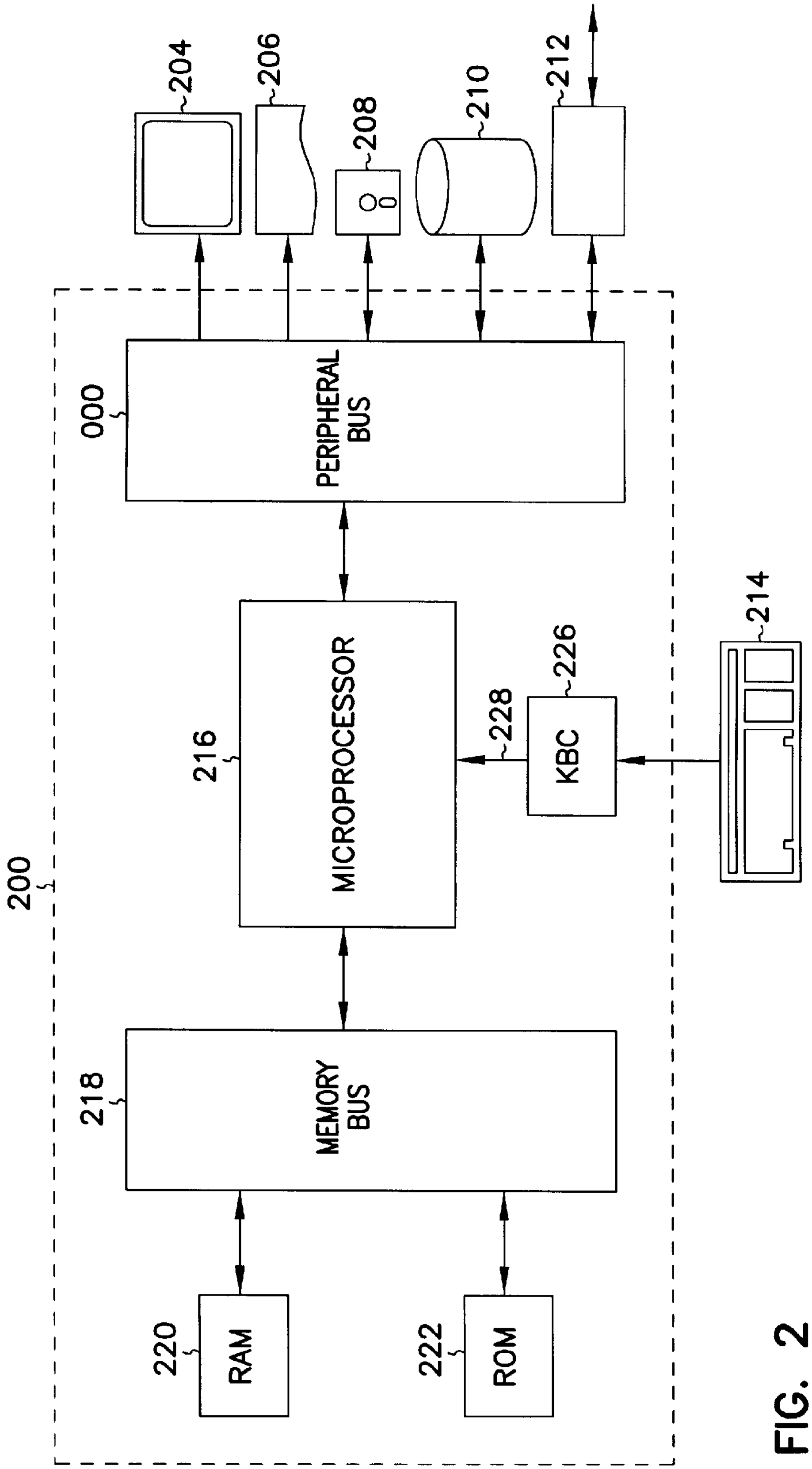

204
206
208
210
212
000
PERIPHERAL BUS
216
MICROPROCESSOR
228
KBC
226
214
200
218
MEMORY BUS
220
RAM
222
ROM

FIG. 2

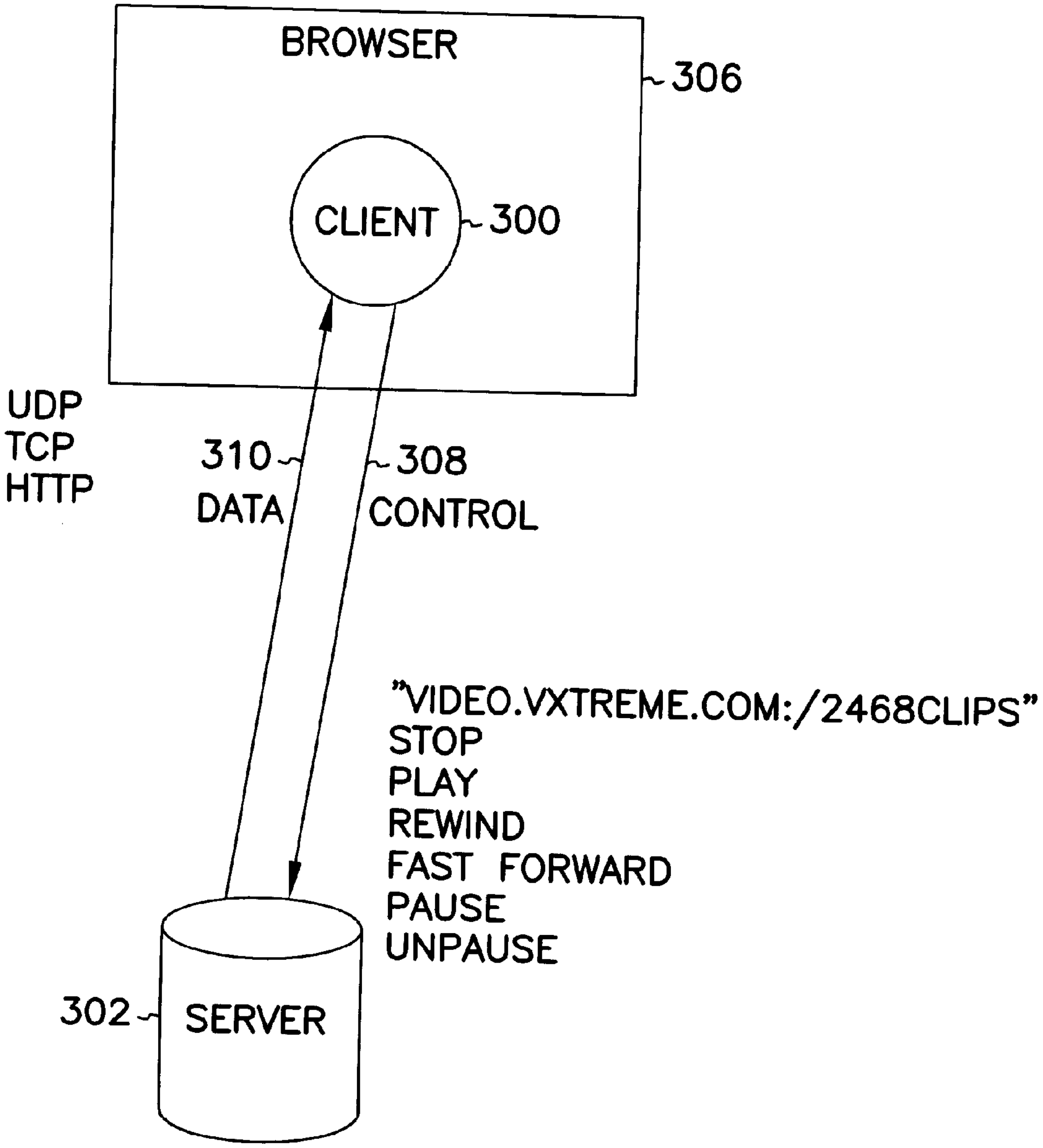

BROWSER
306
CLIENT
300
UDP
TCP
HTTP
310
DATA
308
CONTROL
"VIDEO.VXTREME.COM:/2468CLIPS"
STOP
PLAY
REWIND
FAST FORWARD
PAUSE
UNPAUSE
302
SERVER

FIG. 3

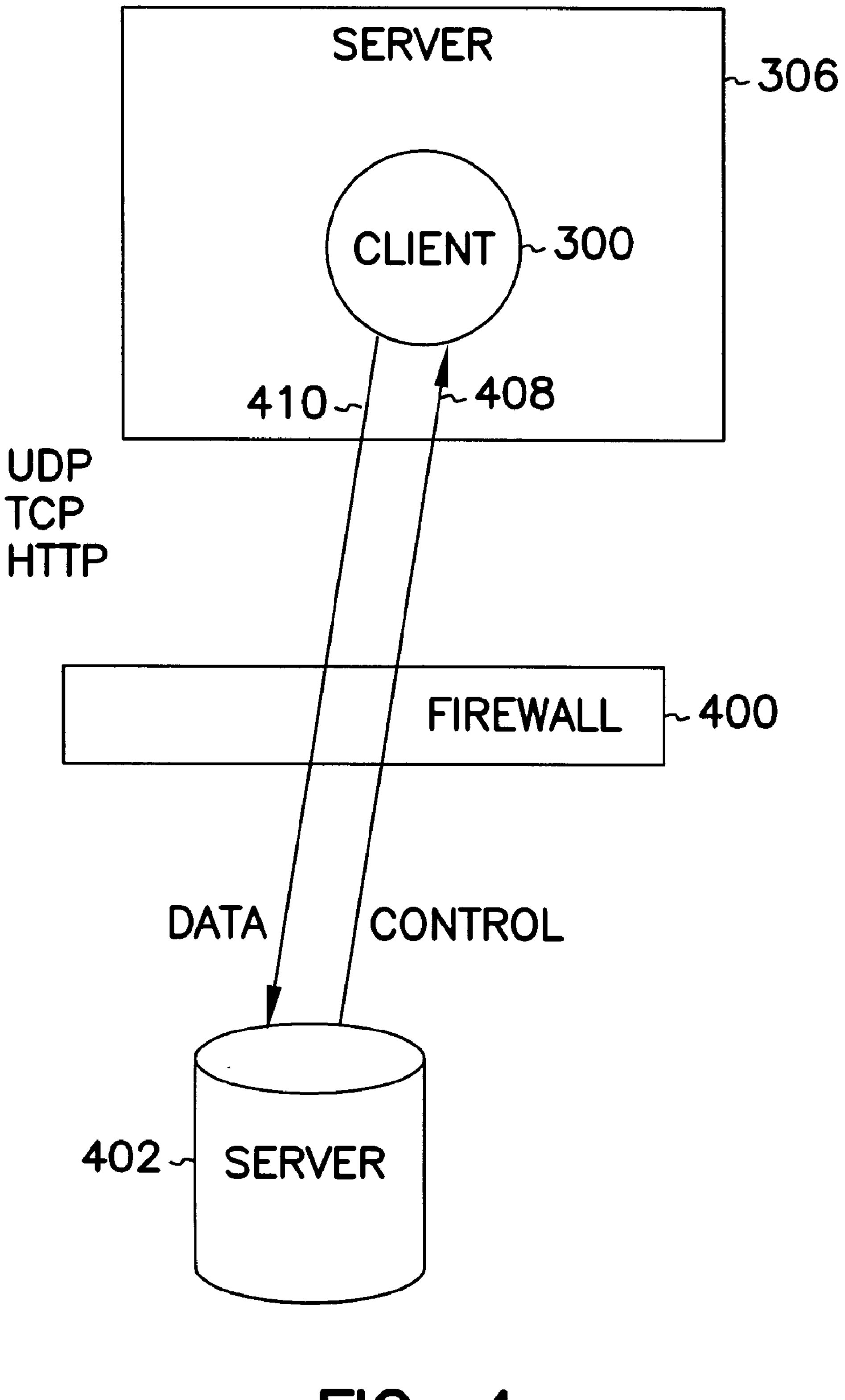

SERVER
306
CLIENT
300
410
408
UDP
TCP
HTTP
FIREWALL
400
DATA
CONTROL
402
SERVER

FIG. 4

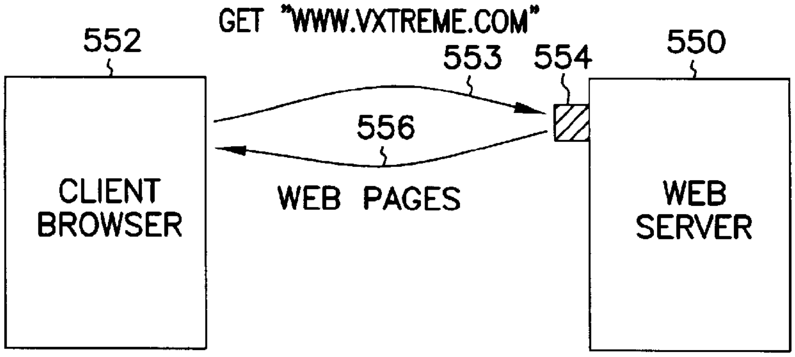


FIG. 5A (PRIOR ART)

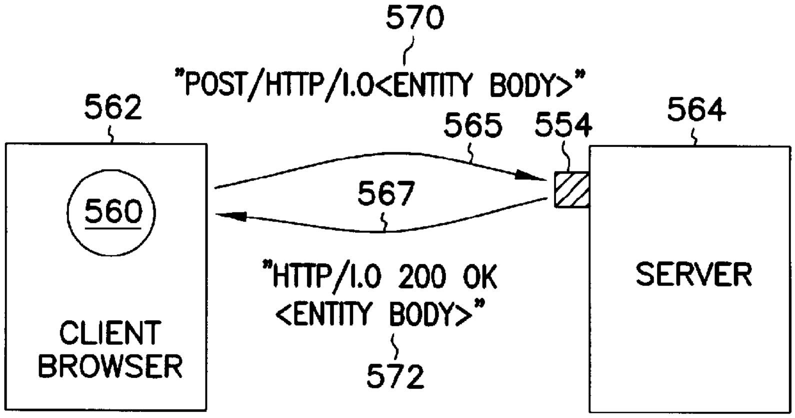


FIG. 5B

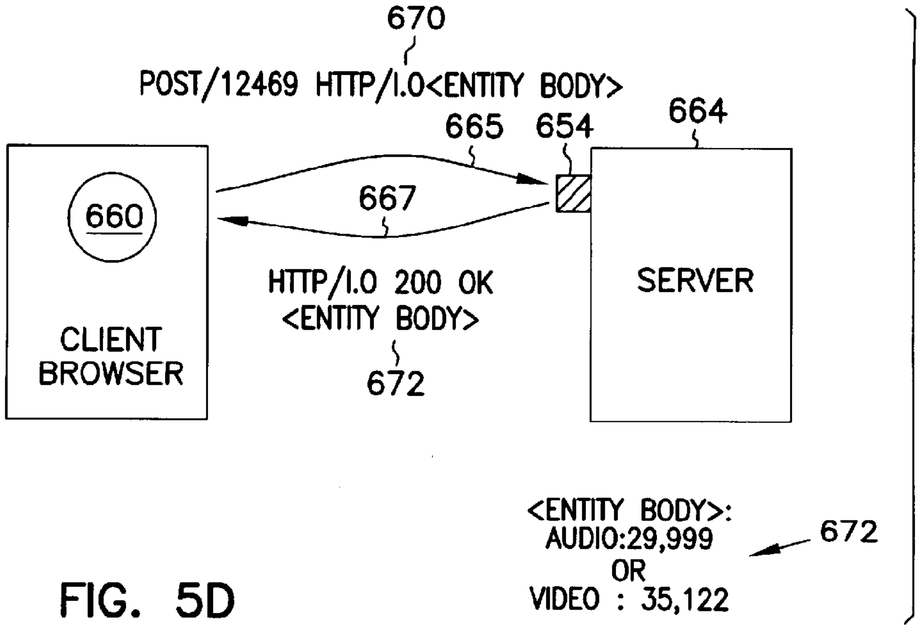

670
POST/12469 HTTP/I.0<ENTITY BODY>
665
654
664
660
667
CLIENT BROWSER
HTTP/I.0 200 OK <ENTITY BODY>
672
SERVER
<ENTITY BODY>: AUDIO:29,999 OR VIDEO : 35,122
672

FIG. 5D

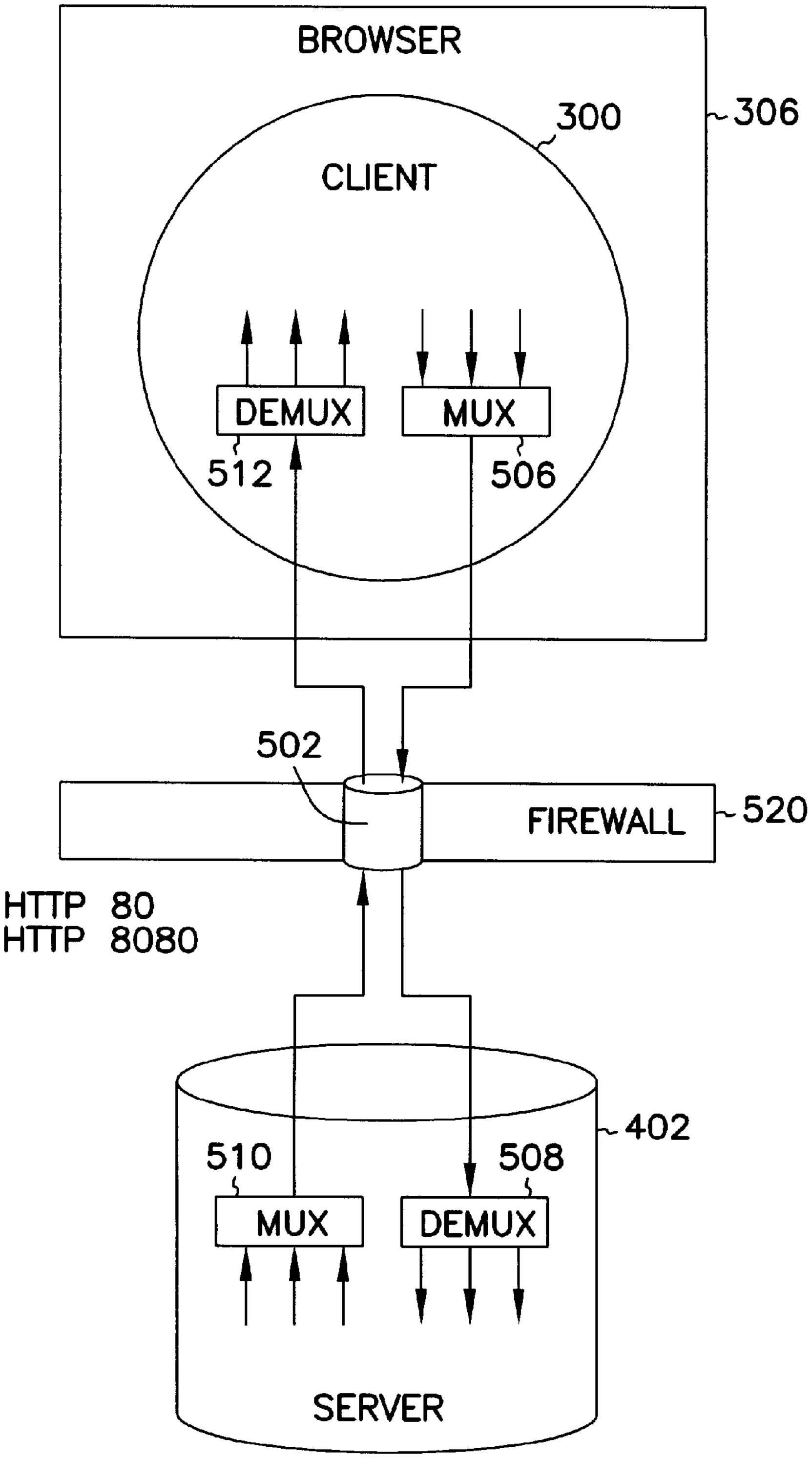

BROWSER
300
306
CLIENT
DEMUX
MUX
512
506
502
FIREWALL
520
HTTP 80
HTTP 8080
402
510
508
MUX
DEMUX
SERVER

FIG. 5C

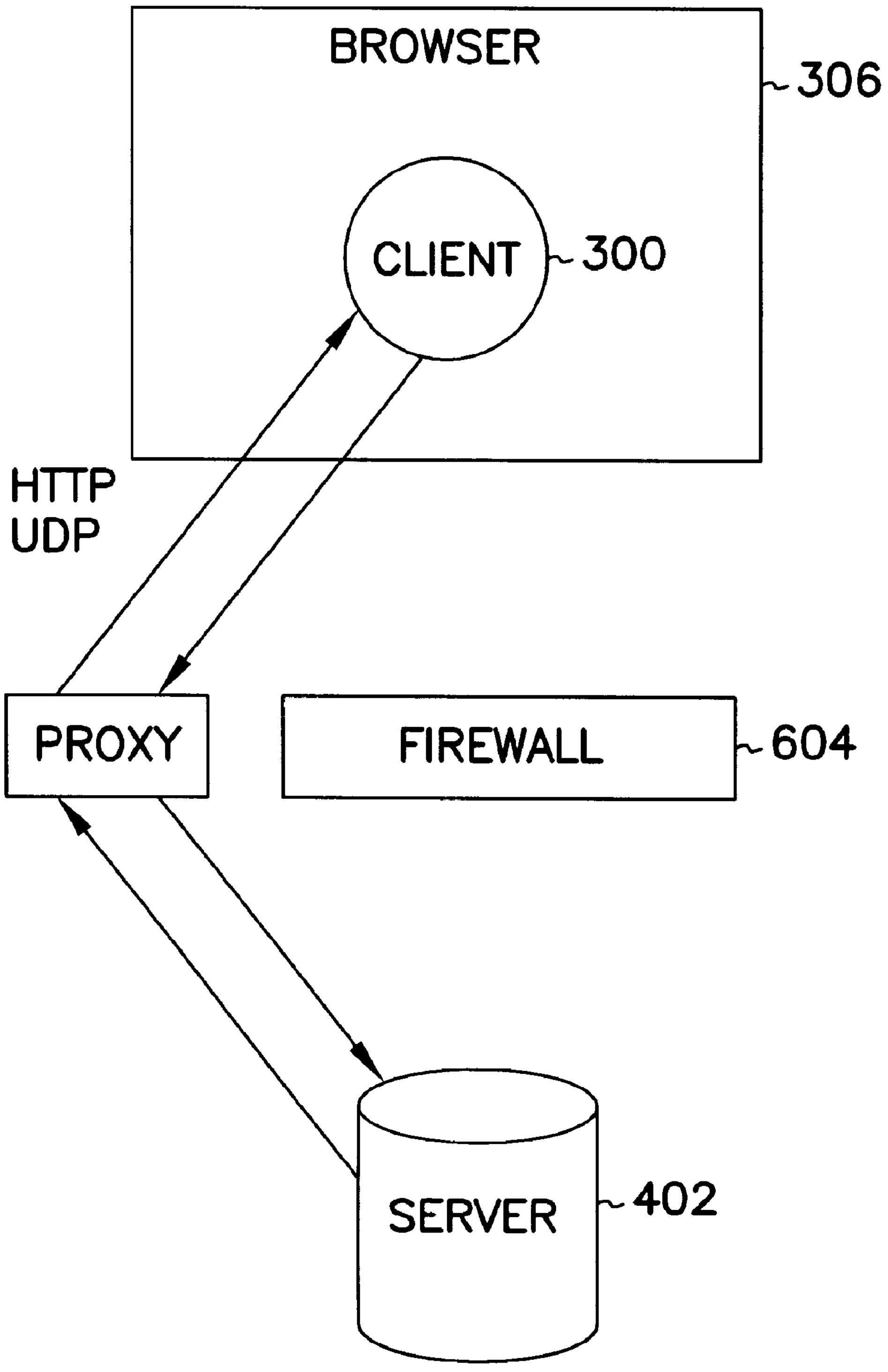

BROWSER
306
CLIENT
300
HTTP
UDP
PROXY
FIREWALL
604
SERVER
402

FIG. 6

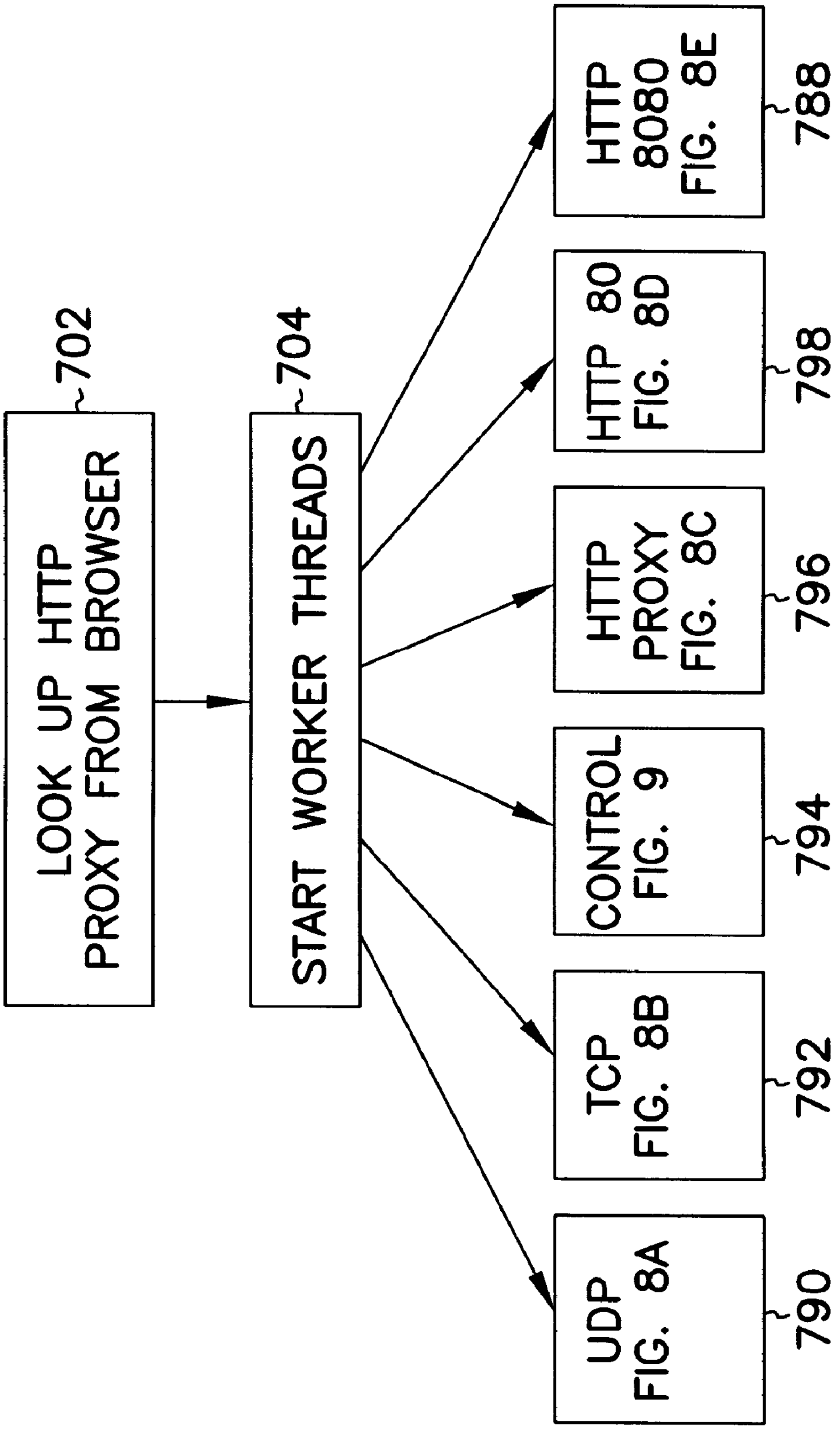

LOOK UP HTTP PROXY FROM BROWSER
702
START WORKER THREADS
704
UDP FIG. 8A
790
TCP FIG. 8B
792
CONTROL FIG. 9
794
HTTP PROXY FIG. 8C
796
HTTP 80 FIG. 8D
798
HTTP 8080 FIG. 8E
788

FIG. 7

FIG. 8A

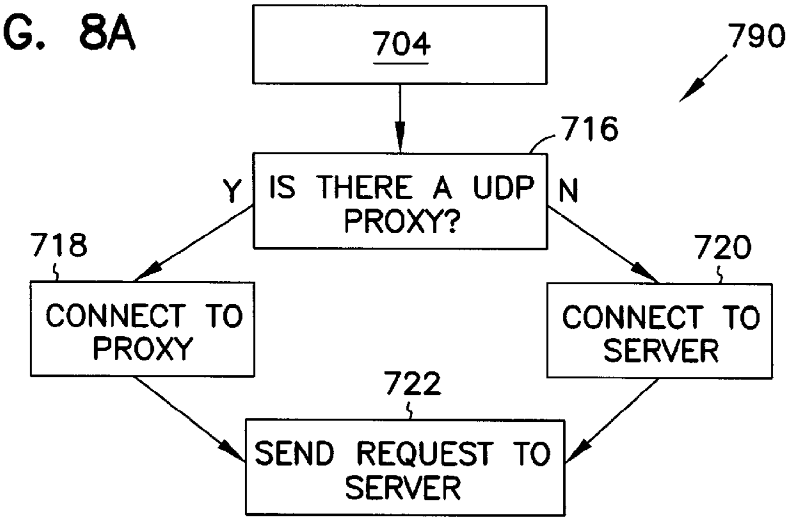


FIG. 8B

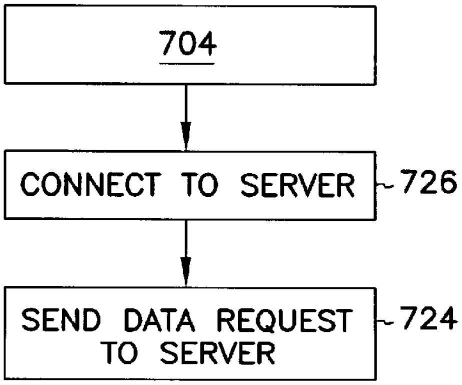


FIG. 8C

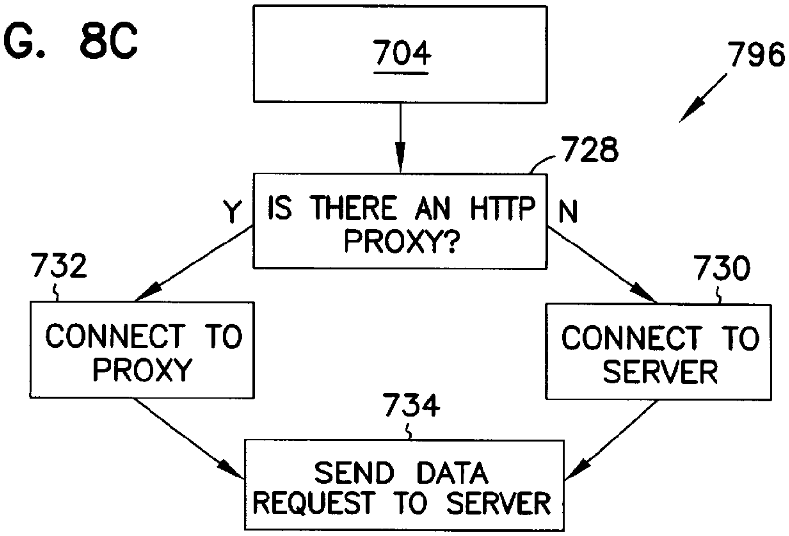


FIG. 8D

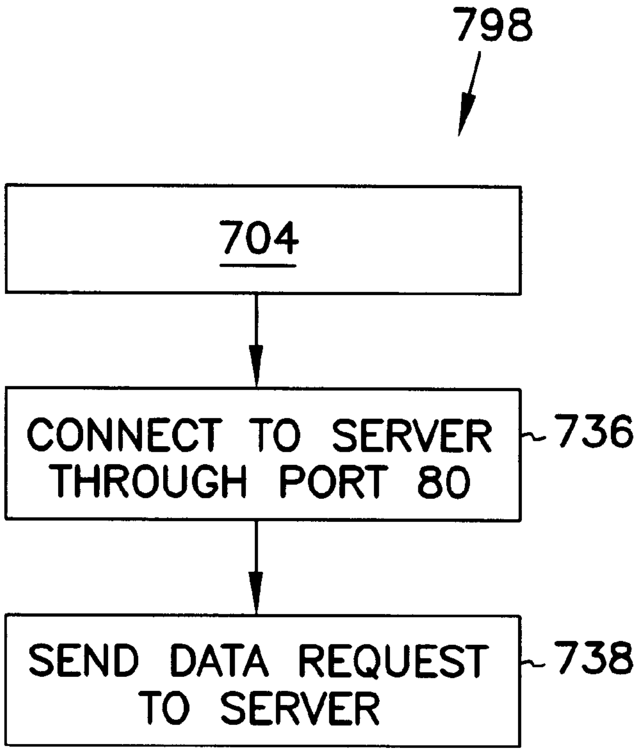
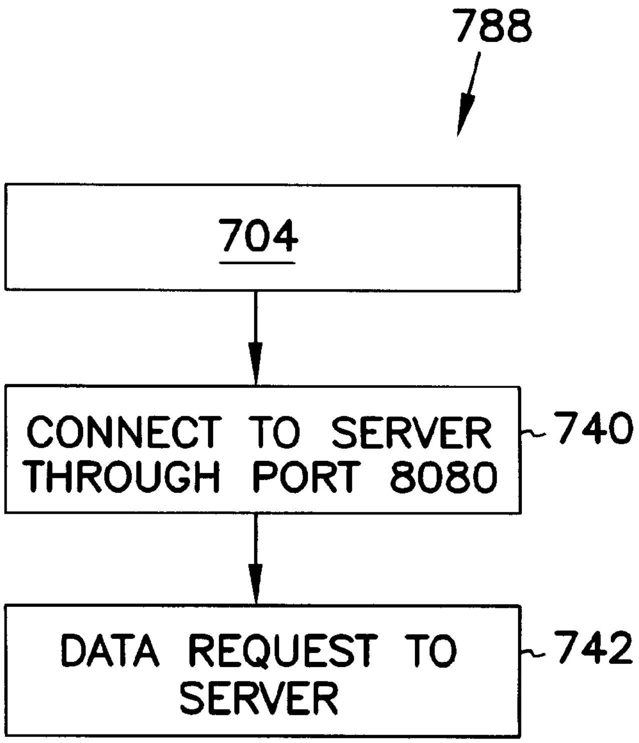

798
704
CONNECT TO SERVER THROUGH PORT 80
736
SEND DATA REQUEST TO SERVER
738

FIG. 8E


788
704
CONNECT TO SERVER THROUGH PORT 8080
740
DATA REQUEST TO SERVER
742

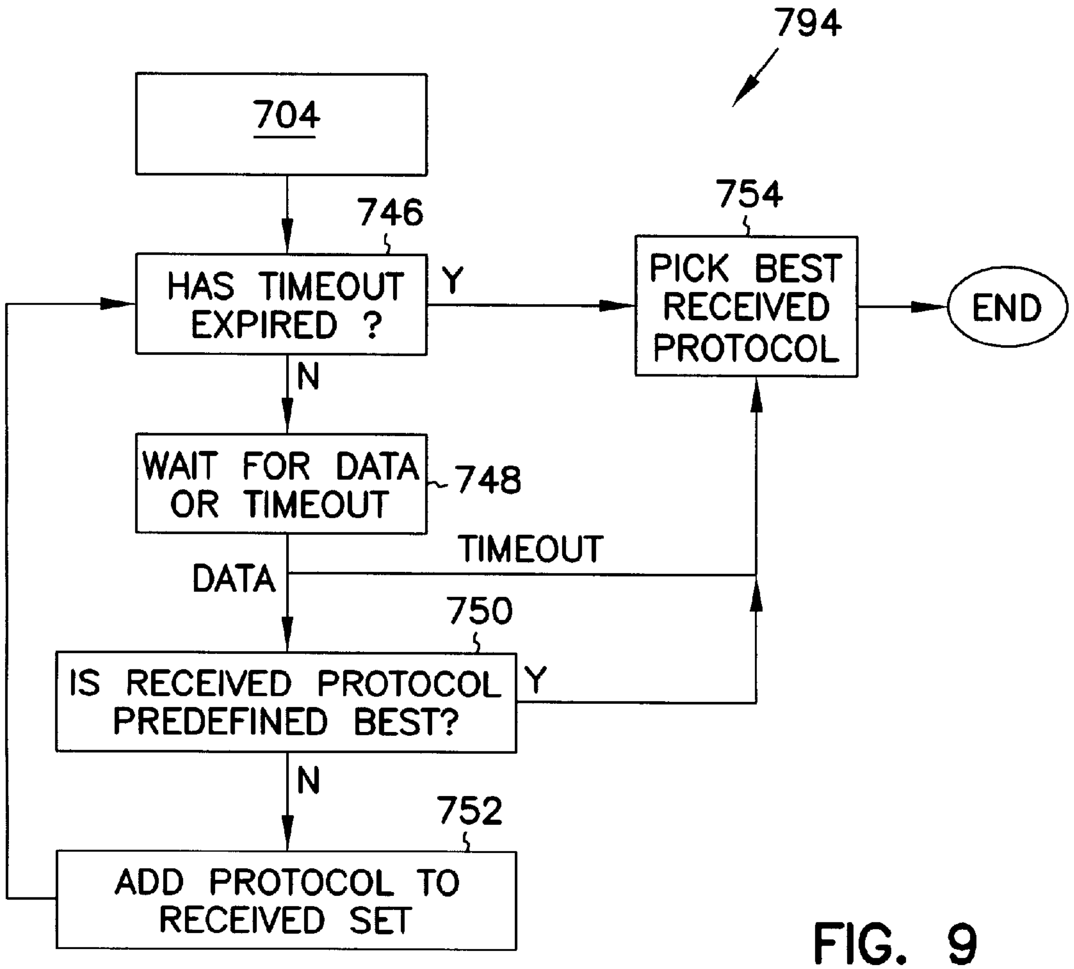

794
704
746
HAS TIMEOUT EXPIRED ?
Y
N
754
PICK BEST RECEIVED PROTOCOL
END
WAIT FOR DATA OR TIMEOUT
748
TIMEOUT
DATA
750
IS RECEIVED PROTOCOL PREDEFINED BEST?
Y
N
752
ADD PROTOCOL TO RECEIVED SET

FIG. 9

METHOD AND APPARATUS FOR DETERMINING A MOST ADVANTAGEOUS PROTOCOL FOR USE IN A COMPUTER NETWORK

This application claims priority under 35 U.S.C 119 (e) of a provisional application entitled "VCR CONTROL FUNCTIONS" filed Jan. 30, 1997 by inventors Anthony W. Cannon, Anders E. Klemets, Hemanth S. Ravi, and David del Val (application Ser. No. 60/036,661) and a provisional application entitled "METHODS AND APPARATUS FOR AUTODETECTING PROTOCOLS IN A COMPUTER NETWORK" filed Jan. 30, 1997 by inventors Anthony W. Cannon, Anders E. Klemets, Hemanth S. Ravi, and David del Val (application Ser. No. 60/036,662).

CROSS REFERENCE TO RELATED APPLICATIONS

This application is related to co-pending U.S. application Ser. No. 08/818,805, filed on Mar. 14, 1997, entitled "Method and Apparatus for Implementing Motion Detection in Video Compression", U.S. application Ser. No. 08/819,507, filed Mar. 14, 1997, entitled "Digital Video Signal Encoder and Encoding Method", U.S. application Ser. No. 08/818,804, filed on Mar. 14, 1997, entitled "Production of a Video Stream with Synchronized Annotations over a Computer Network", U.S. application Ser. No. 08/819,586, filed on Mar. 14, 1997, entitled "Method and Apparatus for Implementing Control Functions in a Streamed Video Display System", U.S. application Ser. No. 08/818,769, filed on Mar. 14, 1997, entitled "Method and Apparatus for Automatically Detecting Protocols in a Computer Network," U.S. application Ser. No.08/818,127, filed on Mar. 14, 1997, entitled "Dynamic Bandwidth Selection for Efficient Transmission of Multimedia Streams in a Computer Network," U.S. application Ser. No. 08/819,585, filed on Mar. 14, 1997, entitled "Streaming and Display of a Video Stream with Synchronized Annotations over a Computer Network", U.S. application Ser. No. 08/818,664, filed on Mar. 14, 1997, entitled "Selective Retransmission for Efficient and Reliable Streaming of Multimedia Packets in a Computer Network", U.S. application Ser. No. 08/819,579, filed Mar. 14, 1997, entitled "Method and Apparatus for Table-Based Compression with Embedded Coding", U.S. application Ser. No. 08/818,826, filed on Mar. 14, 1997, entitled "Digital Video Signal Encoder and Encoding Method", all filed concurrently herewith, U.S. application Ser. No. 08/822,156, filed on Mar. 17, 1997, entitled "Method and Apparatus for Communication Media Commands and Data Using the HTTP Protocol", provisional U.S. application Ser. No. 60/036,662, filed on Jan. 30, 1997, entitled "Methods and Apparatus for Autodetecting Protocols in a Computer Network" U.S. application Ser. No. 08/625,650, filed on Mar. 29, 1996, entitled "Table-Based Low-Level Image Classification System", U.S. application Ser. No. 08/714,447, filed on Sep. 16, 1996, entitled "Multimedia Compression System with Additive Temporal Layers", and is a continuation-in-part of U.S. application Ser. No. 08/623,299, filed on Mar. 28, 1996, entitled "Table-Based Compression with Embedded Coding", which are all incorporated by reference in their entirety for all purposes.

BACKGROUND OF THE INVENTION

The present invention relates to data communication in a computer network. More particularly, the present invention relates to improved methods and apparatus for permitting a

client computer in a client-server architecture computer network to automatically detect the most advantageous protocol, among the protocols available, for use in communicating with the server irrespective whether there exist firewalls or proxies in the network.

Client-server architectures are well known to those skilled in the computer art. For example, in a typical computer network, one or more client computers may be coupled to any number of server computers. Client computers typically refer to terminals or personal computers through which end users interact with the network. Server computers typically represent nodes in the computer network where data, application programs, and the like, reside. Server computers may also represent nodes in the network for forwarding data, programs, and the likes from other servers to the requesting client computers.

To facilitate discussion, FIG. **1** illustrates a computer network **100**, representing for example a subset of an international computer network popularly known as the Internet. As is well known, the Internet represents a well-known international computer network that links, among others, various military, governmental, educational, nonprofit, industrial and financial institutions, commercial enterprises, and individuals. There are shown in FIG. **1** a server **102**, a server **104**, and a client computer **106**. Server computer **104** is separated from client computer **106** by a firewall **108**, which may be implemented in either software or hardware, and may reside on a computer and/or circuit between client computer **106** and server computer **104**.

Firewall **108** may be specified, as is well known to those skilled in the art, to prevent certain types of data and/or protocols from traversing through it. The specific data and/or protocols prohibited or permitted to traverse firewall **108** depend on the firewall parameters, which are typically set by a system administrator responsible for the maintenance and security of client computer **106** and/or other computers connected to it, e.g., other computers in a local area network. By way of example, firewall **108** may be set up to prevent TCP, UDP, or HTTP (Transmission Control Protocol, User Datagram Protocol, and Hypertext Transfer Protocol, respectively) data and/or other protocols from being transmitted between client computer **106** and server **104**. The firewalls could be configured to allow specific TCP or UDP sessions, for example outgoing TCP connection to certain ports, UDP sessions to certain ports, and the like.

Without a firewall, any type of data and/or protocol may be communicated between a client computer and a server computer if appropriate software and/or hardware are employed. For example, server **102** resides on the same side of firewall **108** as client computer **106**, i.e., firewall **108** is not disposed in between the communication path between server **102** and client computer **106**. Accordingly, few, if any, of the protocols that client computer **106** may employ to communicate with server **102** may be blocked.

As is well known to those skilled in the art, some computer networks may be provided with proxies, i.e., software codes or hardware circuitries that facilitate the indirect communication between a client computer and a server around a firewall. With reference to FIG. **1**, for example, client computer **106** may communicate with server **104** through proxy **120**. Through proxy **120**, HTTP data, which may otherwise be blocked by firewall **108** for the purpose of this example, may be transmitted between client computer **106** and server computer **104**.

In some computer networks, one or more protocols may be available for communication between the client computer and the server computer. For certain applications, one of these protocols, however, is often more advantageous, i.e., suitable, than others. By way of example, in applications involving real-time data rendering (such as rendering audio, video, and/or annotation data as they are streamed from a server, as described in above-referenced U.S. patent application Ser. Nos. 08/818,804 and 08/819,585 (Atty: Docket No.: VXT710)), it is highly preferable that the client computer executing that application selects a protocol that permits the greatest degree of control over the transmission of data packets and/or enables data transmission to occur at the highest possible rate. This is because these applications are fairly demanding in terms of their bit rate and connection reliability requirements. Accordingly, the quality of the data rendered, e.g., the video and/or audio clips played, often depends on whether the user has successfully configured the client computer to receive data from the server computer using the most advantageous protocol available.

In the prior art, the selection of the most advantageous protocol for communication between client computer **106** and server computer **104** typically requires a high degree of technical sophistication on the part of the user of client computer **106**. By way of example, it is typically necessary in the prior art for the user of client computer **106** to understand the topology of computer network **100**, the protocols available for use with the network, and/or the protocols that can traverse firewall **108** before that user can be expected to configure his client computer **106** for communication.

This level of technical sophistication is, however, likely to be beyond that typically possessed by an average user of client computer **106**. Accordingly, users in the prior art often find it difficult to configure their client computers even for simple communication tasks with the network. The difficulties may be encountered for example during the initial setup or whenever there are changes in the topology of computer network **100** and/or in the technology employed to transmit data between client computer **106** and server **104**. Typically, expert and expensive assistance is required, if such assistance is available at all in the geographic area of the user.

Furthermore, even if the user can configure client computer **106** to communicate with server **104** through firewall **108** and/or proxy **120**, there is no assurance that the user of client computer **106** has properly selected, among the protocols available, the most advantageous protocol communication (e.g., in terms of data transmission rate, transmission control, and the like). As mentioned earlier, the ability to employ the most advantageous protocol for communication, while desirable for most networking applications, and is particularly critical in applications such as real-time data rendering (e.g., rendering of audio, video, and/or annotation data as they are receive from the server). If a less than optimal protocol is chosen for communication, the quality of the rendered data ,e.g., the video clips and/or audio clips, may suffer.

In view of the foregoing, there are desired improved techniques for permitting a client computer in a client-server network to efficiently, automatically, and appropriately select the most advantageous protocol to communicate with a server computer.

SUMMARY OF THE INVENTION

The invention relates, in one embodiment, to a method in a computer network for automatically detecting a most advantageous protocol for communication by a client computer. The client computer is configured to be coupled to a

server computer via a computer network. The method includes sending from the client computer to the server computer, a plurality of data requests. Each of the data requests employs a different protocol and a different connection. The data requests are configured to solicit, responsive to the data requests, a set of responses from the server computer. Each of the responses employs a protocol associated with a respective one of the data requests.

The method further includes receiving at the client computer at least a subset of the responses. The method also includes monitoring the subset of the responses as each response is received from the server computer to select the most advantageous protocol from protocols associated with the subset of the responses, wherein the most advantageous protocol is determined based on a predefined protocol priority.

In another embodiment, the invention relates to a method in a computer network for automatically detecting a most advantageous protocol for communication by a client computer. The client computer is configured to be coupled to a server computer via a computer network. The method includes sending from the client computer to the server computer, a plurality of data requests. Each of the data requests employs a different protocol and a different connection.

The method further includes receiving at least a subset of the data requests at the server computer. The method additionally includes sending a set of responses from the server computer to the client computer. The set of responses is responsive to the subset of the data requests. Each of the responses employs a protocol associated with a respective one of the subset of the data requests. The method also includes receiving at the client computer at least a subset of the responses. There is further included selecting, for the communication between the client computer and the server computer, the most advantageous protocol from protocols associated with the subset of the responses, wherein the most advantageous protocol is determined based on a predefined protocol priority.

In yet another embodiment, the invention relates to a computer readable medium containing computer-readable instructions for automatically detecting a most advantageous protocol for communication by a client computer. The client computer is configured to be coupled to a server computer via a computer network. The computer-readable instructions comprise computer readable instructions for sending in a substantially parallel manner, from the client computer to the server computer, a plurality of data requests. Each of the data requests employs a different protocol and a different connection. The data requests are configured to solicit, responsive to the data requests, a set of responses from the server computer. Each of the responses employs a protocol associated with a respective one of the data requests.

The computer readable medium further includes computer readable instructions for receiving at the client computer at least a subset of the responses. There is further included computer readable instructions for monitoring the subset of the responses as each response is received from the server computer to select the most advantageous protocol from protocols associated with the subset of the responses, wherein the most advantageous protocol is determined based on a predefined protocol priority.

These and other features of the present invention will be described in more detail below in the detailed description of the invention and in conjunction with the following figures.

BRIEF DESCRIPTION OF THE DRAWINGS

To facilitate discussion, FIG. **1** illustrates a computer network, representing for example a portion of an international computer network popularly known as the Internet.

FIG. **2** is a block diagram of an exemplar computer system for carrying out the autodetect technique according to one embodiment of the invention.

FIG. **3** illustrates, in accordance with one embodiment, the control and data connections between a client application and a server computer when no firewall is provided in the network.

FIG. **4** illustrates another network arrangement wherein control and data connections are established through a firewall.

FIGS. **5**A–B illustrates another network arrangement wherein media control commands and media data may be communicated between a client computer and a server computer using the HTTP protocol.

FIGS. **5**C–D illustrate another network arrangement wherein multiple HTTP control and data connections are multiplexed through a single HTTP port.

FIG. **6** illustrates another network arrangement wherein control and data connections are transmitted between the client application and the server computer via a proxy.

FIG. **7** depicts, in accordance with one embodiment of the present invention, a simplified flowchart illustrating the steps of the inventive autodetect technique.

FIG. **8**A depicts, in accordance with one aspect of the present invention, the steps involved in executing the UDP protocol thread of FIG. **7**.

FIG. **8**B depicts, in accordance with one aspect of the present invention, the steps involved in executing the TCP protocol thread of FIG. **7**.

FIG. **8**C depicts, in accordance with one aspect of the present invention, the steps involved in executing the HTTP protocol thread of FIG. **7**.

FIG. **8**D depicts, in accordance with one aspect of the present invention, the steps involved in executing the HTTP **80** protocol thread of FIG. **7**.

FIG. **8**E depicts, in accordance with one aspect of the present invention, the steps involved in executing the HTTP **8080** protocol thread of FIG. **7**.

FIG. **9** illustrates, in accordance with one embodiment of the present invention, the steps involved in executing the control thread of FIG. **7**.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention will now be described in detail with reference to a few preferred embodiments thereof as illustrated in the accompanying drawings. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without some or all of these specific details. In other instances, well known process steps have not been described in detail in order to not unnecessarily obscure the present invention.

In accordance with one aspect of the present invention, the client computer in a heterogeneous client-server computer network (e.g., client computer **106** in FIG. **1**) is provided with an autodetect mechanism. When executed, the autodetect mechanism advantageously permits client computer **106** to select, in an efficient and automatic manner, the most advantageous protocol for communication between the client computer and its server. Once the most advantageous protocol is selected, parameters pertaining to the selected protocol are saved to enable the client computer, in future sessions, to employ the same selected protocol for communication.

In accordance with one particular advantageous embodiment, the inventive autodetect mechanism simultaneously employs multiple threads, through multiple connections, to initiate communication with the server computer, e.g., server **104**. Each thread preferably employs a different protocol and requests the server computer to respond to the client computer using the protocol associated with that thread. For example, client computer **106** may, employing the autodetect mechanism, initiate five different threads, using respectively the TCP, UDP, one of HTTP and HTTP proxy, HTTP through port (multiplex) **80**, and HTTP through port (multiplex) **8080** protocols to request server **104** to respond.

Upon receiving a request, server **104** responds with data using the same protocol as that associated with the thread on which the request arrives. If one or more protocols is blocked and fails to reach server **104** (e.g., by a firewall), no response employing the blocked protocol would of course be transmitted from server **104** to client computer **106**. Further, some of the protocols transmitted from server **104** to client computer **106** may be blocked as well. Accordingly, client computer may receive only a subset of the responses sent from server **104**.

In one embodiment, client computer **106** monitors the set of received responses. If the predefined "best" protocol is received, that protocol is then selected for communication by client computer **106**. The predefined "best" protocol may be defined in advance by the user and/or the application program. If the predefined "best" protocol is, however, blocked (as the request is transmitted from the client computer or as the response is transmitted from the server, for example) the most advantageous protocol may simply be selected from the set of protocols received back by the client computer. In one embodiment, the selection may be made among the set of protocols received back by the client computer within a predefined time period after the requests are sent out in parallel.

The selection of the most advantageous protocol for communication among the protocols received by client computer **106** may be performed in accordance with some predefined priority. For example, in the real-time data rendering application, the UDP protocol may be preferred over TCP protocol, which may be in turned preferred over the HTTP protocol. This is because UDP protocol typically can handle a greater data transmission rate and may allow client computer **106** to exercise a greater degree of control over the transmission of data packets.

HTTP data, while popular nowadays for use in transmitting web pages, typically involves a higher number of overhead bits, making it less efficient relative to the UDP protocol for transmitting real-time data. As is known, the HTTP protocol is typically built on top of TCP. The underlaying TCP protocol typically handles the transmission and retransmission requests of individual data packets automatically. Accordingly, the HTTP protocol tends to reduce the degree of control client computer **106** has over the transmission of the data packets between server **104** and client computer **106**. Of course other priority schemes may exist for different applications, or even for different real-time data rendering applications.

In one embodiment, as client computer **106** is installed and initiated for communication with server **104** for the first time, the autodetect mechanism is invoked to allow client computer **106** to send transmission requests in parallel (e.g., using different protocols over different connections) in the manner discussed earlier. After server **104** responds with data via multiple connections/protocols and the most advantageous protocol has been selected by client computer **106** for communication (in accordance with some predefined priority), the parameters associated with the selected protocol are then saved for future communication.

Once the most advantageous protocol is selected, the autodetect mechanism may be disabled, and future communication between client computer **106** and server **104** may proceed using the selected most advantageous protocol without further invocation of the autodetect mechanism. If the topology of computer network **100** changes and communication using the previously selected "most advantageous" protocol is no longer appropriate, the autodetect mechanism may be executed again to allow client computer **106** to ascertain a new "most advantageous" protocol for communication with server **104**. In one embodiment, the user of client computer **106** may, if desired, initiate the autodetect mechanism at anytime in order to enable client computer **106** to update the "most advantageous" protocol for communication with server **104** (e.g., when the user of client computer **106** has reasons to suspect that the previously selected "most advantageous" protocol is no longer the most optimal protocol for communication).

The inventive autodetect mechanism may be implemented either in software or hardware, e.g., via an IC chip. If implemented in software, it may be carried out by any number of computers capable of functioning as a client computer in a computer network. FIG. **2** is a block diagram of an exemplar computer system **200** for carrying out the autodetect technique according to one embodiment of the invention. Computer system **200**, or an analogous one, may be employed to implement either a client or a server of a computer network. The computer system **200** includes a digital computer **202**, a display screen (or monitor) **204**, a printer **206**, a floppy disk drive **208**, a hard disk drive **210**, a network interface **212**, and a keyboard **214**. The digital computer **202** includes a microprocessor **216**, a memory bus **218**, random access memory (RAM) **220**, read only memory (ROM) **222**, a peripheral bus **224**, and a keyboard controller **226**. The digital computer **200** can be a personal computer (such as an Apple computer, e.g., an Apple Macintosh, an IBM personal computer, or one of the compatibles thereof), a workstation computer (such as a Sun Microsystems or Hewlett-Packard workstation), or some other type of computer.

The microprocessor **216** is a general purpose digital processor which controls the operation of the computer system **200**. The microprocessor **216** can be a single-chip processor or can be implemented with multiple components. Using instructions retrieved from memory, the microprocessor **216** controls the reception and manipulation of input data and the output and display of data on output devices.

The memory bus **218** is used by the microprocessor **216** to access the RAM **220** and the ROM **222**. The RAM **220** is used by the microprocessor **216** as a general storage area and as scratch-pad memory, and can also be used to store input data and processed data. The ROM **222** can be used to store instructions or program code followed by the microprocessor **216** as well as other data.

The peripheral bus **224** is used to access the input, output, and storage devices used by the digital computer **202**. In the described embodiment, these devices include the display screen **204**, the printer device **206**, the floppy disk drive **208**, the hard disk drive **210**, and the network interface **212**, which is employed to connect computer **200** to the network. The keyboard controller **226** is used to receive input from keyboard **214** and send decoded symbols for each pressed key to microprocessor **216** over bus **228**.

The display screen **204** is an output device that displays images of data provided by the microprocessor **216** via the peripheral bus **224** or provided by other components in the computer system **200**. The printer device **206** when operating as a printer provides an image on a sheet of paper or a similar surface. Other output devices such as a plotter, typesetter, etc. can be used in place of, or in addition to, the printer device **206**.

The floppy disk drive **208** and the hard disk drive **210** can be used to store various types of data. The floppy disk drive **208** facilitates transporting such data to other computer systems, and hard disk drive **210** permits fast access to large amounts of stored data.

The microprocessor **216** together with an operating system operate to execute computer code and produce and use data. The computer code and data may reside on the RAM **220**, the ROM **222**, the hard disk drive **220**, or even on another computer on the network. The computer code and data could also reside on a removable program medium and loaded or installed onto the computer system **200** when needed. Removable program mediums include, for example, CD-ROM, PC-CARD, floppy disk and magnetic tape.

The network interface circuit **212** is used to send and receive data over a network connected to other computer systems. An interface card or similar device and appropriate software implemented by the microprocessor **216** can be used to connect the computer system **200** to an existing network and transfer data according to standard protocols.

The keyboard **214** is used by a user to input commands and other instructions to the computer system **200**. Other types of user input devices can also be used in conjunction with the present invention. For example, pointing devices such as a computer mouse, a track ball, a stylus, or a tablet can be used to manipulate a pointer on a screen of a general-purpose computer.

The invention can also be embodied as computer readable code on a computer readable medium. The computer readable medium is any data storage device that can store data which can be thereafter be read by a computer system. Examples of the computer readable medium include read-only memory, random-access memory, CD-ROMs, magnetic tape, optical data storage devices. The computer readable code can also be distributed over a network coupled computer systems so that the computer readable code is stored and executed in a distributed fashion.

FIGS. **3**–**6** below illustrate, to facilitate discussion, some possible arrangements for the transmission and receipt of data in a computer network. The arrangements differ depend on which protocol is employed and the configuration of the network itself. FIG. **3** illustrates, in accordance with one embodiment, the control and data connections between a client application **300** and server **302** when no firewall is provided in the network.

Client application **300** may represent, for example, the executable codes for executing a real-time data rendering program such as the Web Theater Client 2.0, available from VXtreme, Inc. of Sunnyvale, Calif. In the example of FIG. **3**, client application **300** includes the inventive autodetect mechanism and may represent a plug-in software module that may be installed onto a browser **306**. Browser **306** may represent, for example, the application program which the user of the client computer employs to navigate the network. By way of example, browser **306** may represent one of the popular Internet browser programs, such as Netscape™ by

Netscape Communications Inc. of Mountain View, Calif. or Microsoft Explorer by Microsoft Corporation of Redmond, Wash.

When the autodetect mechanism of client application **300** is executed in browser **306** (e.g., during the set up of client application **300**), client application **300** sends a control request over control connection **308** to server **302**. Although multiple control requests are typically sent in parallel over multiple control connections using different protocols as discussed earlier, only one control request is depicted in FIG. **3** to facilitate ease of illustration.

The protocol employed to send the control request over control connection **308** may represent, for example, TCP, or HTTP. If UDP protocol is requested from the server, the request from the client may be sent via the control connection using for example the TCP protocol. Initially, each control request from client application **300** may include, for example, the server name that identifies server **302**, the port through which control connection may be established, and the name of the video stream requested by client application **300**. Server **302** then responds with data via data connection **310**.

In FIG. **3**, it is assumed that no proxies and/or firewalls exist. Accordingly, server **302** responds using the same protocol as that employed in the request. If the request employs TCP, however, server **302** may attempt to respond using either UDP or TCP data connections (depending on the specifics of the request). The response is sent to client application via data connection **310**. If the protocol received by the client application is subsequently selected to be the "most advantageous" protocol, subsequent communication between client application **300** and server **302** may take place via control connection **308** and data connection **310**. Subsequent control requests sent by client application **300** via control connection **308** may include, for example, stop, play, fast forward, rewind, pause, unpause, and the like. These control requests may be utilized by server **302** to control the delivery of the data stream from server **302** to client application **300** via data connection **310**.

It should be noted that although only one control connection and one data connection is shown in FIG. **3** to simplify illustration, multiple control and data connections utilizing the same protocol may exist during a data rendering session. Multiple control and data connections may be required to handle the multiple data streams (e.g., audio, video, annotation) that may be needed in a particular data rendering session. If desired, multiple clients applications **300** may be installed within browser **306**, e.g., to simultaneously render multiple video clips, each with its own sound and annotations.

FIG. **4** illustrates another network arrangement wherein control and data connections are established through a firewall. As mentioned earlier, a firewall may have policies that restrict or prohibit the traversal of certain types of data and/or protocols. In FIG. **4**, a firewall **400** is disposed between client application **300** and server **402**. Upon execution, client application **300** sends control request using a given protocol via firewall **400** to server **402**. Server **402** then responds with data via data connection **410**, again via firewall **400**.

If the data and/or protocol can be received by the client computer through firewall **400**, client application **300** may then receive data from server **402** (through data connection **408**) in the same protocol used in the request. As before, if the request employs the TCP protocol, the server may respond with data connections for either TCP or UDP protocol (depending on the specifics of the request). Protocols that may traverse a firewall may include one or more of the following: UDP, TCP, and HTTP.

In accordance with one aspect of the present invention, the HTTP protocol may be employed to send/receive media data (video, audio, annotation, or the like) between the client and the server. FIG. **5A** is a prior art drawing illustrating how a client browser may communicate with a web server using a port designated for communication. In FIG. **5**, there is shown a web server **550**, representing the software module for serving web pages to a browser application **552**. Web server **550** may be any of the commercially available web servers that are available from, for example, Netscape Communications Inc. of Mountain View, Calif. or Microsoft Corporation of Redmond, Wash. Browser application **552** represents for example the Netscape browser from the aforementioned Netscape Communications, Inc., or similarly suitable browser applications.

Through browser application **552**, the user may, for example, obtain web pages pertaining to a particular entity by sending an HTTP request (e.g., GET) containing the URL (uniform resource locator) that identifies the web page file. The request sent via control connection **553** may arrive at web server **550** through the HTTP port **554**. HTTP port **554** may represent any port through which HTTP communication is enabled. HTTP port **554** may also represent the default port for communicating web pages with client browsers. The HTTP default port may represent, for example, either port **80** or port **8080** on web server **550**. As is known, one or both of these ports on web server **550** may be available for web page communication even if there are firewalls disposed between the web server **550** and client browser application **552**, which otherwise block all HTTP traffic in other ports. Using the furnished URL, web server **550** may then obtain the desired web page(s) for sending to client browser application **552** via data connection **556**.

The invention, in one embodiment, involves employing the HTTP protocol to communicate media commands from a browser application or browser plug-in to the server. Media commands are, for example, PLAY, STOP, REWIND, FAST FORWARD, and PAUSE. The server computer may represent, for example, a web server. The server computer may also represent a video server for streaming video to the client computer. Through the use of the HTTP protocol the client computer may successfully send media control requests and receive media data through any HTTP port. If the default HTTP port, e.g., port **80** or **8080**, is specified, the client may successfully send media control requests and receive media data even if there exists a firewall or an HTTP Proxy disposed in between the server computer and the client computer, which otherwise blocks all other traffic that does not use the HTTP protocol. For example, these firewalls or HTTP Proxies do not allow regular TCP or UDP packets to go through.

As is well known to those skilled, the HTTP protocol, as specified by for example the Internet Request For Comments RFC 1945 (T. Berners-Lee et al.), typically defines only three types of requests to be sent from the client computer to the server, namely GET, POST, and HEAD. The POST command, for instance, is specified in RFC 1945 to be composed of a Request-Line, one or more Headers and Entity-Body. To send media commands like PLAY, REWIND, etc., the invention in one embodiment sends the media command as part of the Entity-Body of the HTTP POST command. The media command can be in any format or protocol, and can be, for instance, in the same format as that employed when firewalls are not a concern and plain

TCP protocol can be used. This format can be, for example, RTSP (Real Time Streaming Protocol).

When a server gets an HTTP request, it answers the client with an HTTP Response. Responses are typically composed of a Status-Line, one or more headers, and an Entity-Body. In one embodiment of this invention, the response to the media commands is sent as the Entity-Body of the response to the original HTTP request that carried the media command.

FIG. **5B** illustrates this use of HTTP for sending arbitrary media commands. In FIG. **5B**, the plug-in application **560** within client browser application **562** may attempt to receive media data (e.g., video, audio, annotation, or the like) by first sending an HTTP request to server **564** via control connection **565**. For example, a REWIND command could be sent from the client **560** to the server **564** as an HTTP packet **570** of the form: "POST/HTTP/1.0<Entity-Body containing rewind command in any suitable media protocol>". The server can answer to this request with an HTTP response **572** of the form: "HTTP/1.0 200ok<Entity-Body containing rewind response in any suitable media protocol>".

The HTTP protocol can be also used to send media data across firewalls. The client can send a GET request to the video server, and the video server can then send the video data as the Entity-Body of the HTTP response to this GET request.

Some firewalls may be restrictive with respect to HTTP data and may permit HTTP packets to traverse only on a certain port, e.g., port **80** and/or port **8080**. FIG. **5C** illustrates one such situation. In this case, the control and data communications for the various data stream, e.g., audio, video, and/or annotation associated with different rendering sessions (and different clients) may be multiplexed using conventional multiplexer code and/or circuit **506** at client application **300** prior to being sent via port **502** (which may represent, for example, HTTP port **80** or HTTP port **8080**). The inventive combined use of the HTTP protocol and of the multiplexer for transmitting media control and data is referred to as the HTTP multiplex protocol, and can be used to send this data across firewalls that only allow HTTP traffic on specific ports, e.g., port **80** or **8080**.

At server **402**, representing, for example, server **104** of FIG. **1**, conventional demultiplexer code and/or circuit **508** may be employed to decode the received data packets to identify which stream the control request is associated with. Likewise, data sent from server **402** to client application **300** may be multiplexed in advance at server **402** using for example conventional multiplexer code and/or circuit **510**. The multiplexed data is then sent via port **502**. At client application **300**, the multiplexed data may be decoded via conventional demultiplexer code and/or circuit **512** to identify which stream the received data packets is associated with (audio, video, or annotation).

Multiplexing and demultiplexing at the client and/or server may be facilitated for example by the use of the Request-URL part of the Request-Line of HTTP requests. As mentioned above, the structure of HTTP requests is described in RFC **1945**. The Request-URL may, for example, identify the stream associated with the data and/or control request being transmitted. In one embodiment, the additional information in the Request-URL in the HTTP header may be as small as one or a few bits added to the HTTP request sent from client application **300** to server **402**.

To further facilitate discussion of the inventive HTTP multiplexing technique, reference may now be made to FIG. **5D**. In FIG. **5D**, the plug-in application **660** within client plug-in application **660** may attempt to receive media data (e.g., video, audio, annotation, or the like) by first sending a control request **670** to server **664** via control connection **665**. The control request is an HTTP request, which arrives at the HTTP default port **654** on server **664**. As mentioned earlier, the default HTTP port may be either port **80** or port **8080** in one embodiment.

In one example, the control request **670** from client plug-in **660** takes the form of a command to "POST/12469 HTTP/1.0<Entity-Body>" which indicates to the server (through the designation 12469 as the Request-URL) that this is a control connection. The Entity-Body contains, as described above, binary data that informs the video server that the client plug-in **660** wants to display a certain video or audio clip. Software codes within server **664** may be employed to assign a unique ID to this particular request from this particular client.

For discussion sake, assume that server **664** associates unique ID 35,122 with a video data connection between itself and client plug-in application **660**, and unique ID 29,999 with an audio data connection between itself and client plug-in application. The unique ID is then communicated as message **672** from server **664** to client plug-in application **660**, again through the aforementioned HTTP default port using data connection **667**. The Entity-Body of message **672** contains, among other things and as depicted in detail **673**, the audio and/or video session ID. Note that the unique ID is unique to each data connection (e.g., each of the audio, video, and annotation connections) of each client plug-in application (since there may be multiple client plug-in applications attempting to communicate through the same port).

Once the connection is established, the same unique ID number is employed by the client to issue HTTP control requests to server **664**. By way of example, client plug-in application **660** may issue a command "GET /35,122 HTTP/1.0" or "POST /35,122 HTTP/1.0<Entity-Body containing binary data with the REWIND media command>" to request a video file or to rewind on the video file. Although the rewind command is used in FIGS. **5A**–**5D** to facilitate ease of discussion, other media commands, e.g., fast forward, pause, real-time play, live-play, or the like, may of course be sent in the Entity-Body. Note that the unique ID is employed in place of or in addition to the Request-URL to qualify the Request-URL.

Once the command is received by server **664**, the unique ID number (e.g. 35,122) may be employed by the server to demultiplex the command to associate the command with a particular client and data file. This unique ID number can also attach to the HTTP header of HTTP responses sent from server **664** to client plug-in application **660**, through the same HTTP default port **654** on server **664**, to permit client plug-in application **660** to ascertain whether an HTTP data packet is associated with a given data stream.

Advantageously, the invention permits media control commands and media data to be communicated between the client computer and the server computer via the default HTTP port, e.g., port **80** or **8080** in one embodiment, even if HTTP packets are otherwise blocked by a firewall disposed between the client computer and the server computer. The association of each control connection and data connection to each client with a unique ID advantageously permits multiple control and data connections (from one or more clients) to be established through the same default HTTP port on the server, advantageously bypassing the firewall. Since both the server and the client have the demultiplexer code and/or circuit that resolve a particular unique ID into a particular data stream, multiplexed data communication is advantageously facilitated thereby.

In some networks, it may not be possible to traverse the firewall due to stringent firewall policies. As mentioned earlier, it may be possible in these situations to allow the client application to communicate with a server using a proxy. FIG. **6** illustrates this situation wherein client application **300** employs proxy **602** to communicate with server **402**. The use of proxy **602** may be necessary since client application **300** may employ a protocol which is strictly prohibited by firewall **604**. The identity of proxy **602** may be found in browser program **306**, e.g., Netscape as it employs the proxy to download its web pages, or may be configured by the user himself. Typical protocols that may employ a proxy for communication, e.g., proxy **602**, includes HTTP and UDP.

In accordance with one embodiment of the present invention, the multiple protocols that may be employed for communication between a server computer and a client computer are tried in parallel during autodetect. In other words, the connections depicted in FIGS. **3**, **4**, **5**C, and **6** may be attempted simultaneously and in parallel over different control connections by the client computer. Via these control connections, the server is requested to respond with various protocols.

If the predefined "best" protocol (predetermined in accordance with some predefined protocol priority) is received by the client application from the server, autodetect may, in one embodiment, end immediately and the "best" protocol is selected for immediate communication. In one real-time data rendering application, UDP is considered the "best" protocol, and the receipt of UDP data by the client may trigger the termination of the autodetect.

If the "best" protocol has not been received after a predefined time period, the most advantageous protocol (in terms of for example data transfer rate and/or transmission control) is selected among the set of protocols received by the client. The selected protocol may then be employed for communication between the client and the server.

FIG. **7** depicts, in accordance with one embodiment of the present invention, a simplified flowchart illustrating the steps of the inventive autodetect technique. In FIG. **7**, the client application starts (in step **702**) by looking up the HTTP proxy, if there is any, from the browser. As stated earlier, the client computer may have received a web page from the browser, which implies that the HTTP protocol may have been employed by the browser program for communication. If a HTTP proxy is required, the name and location of the HTTP proxy is likely known to the browser, and this knowledge may be subsequently employed by the client to at least enable communication with the server using the HTTP proxy protocol, i.e., if a more advantageous protocol cannot be ascertained after autodetect.

In step **704**, the client begins the autodetect sequence by starting in parallel the control thread **794**, along with five protocol threads **790**, **792**, **796**, **798**, and **788**. As the term is used herein, parallel refers to both the situation wherein the multiple protocol threads are sent parallelly starting at substantially the same time (having substantially similar starting time), and the situation wherein the multiple protocol threads simultaneously execute (executing at the same time), irrespective when each protocol thread is initiated. In the latter case, the multiple threads may have, for example, staggered start time and the initiation of one thread may not depend on the termination of another thread.

Control tread **794** represents the thread for selecting the most advantageous protocol for communication. The other protocol threads **790**, **792**, **796**, **798**, and **788** represent threads for initiating in parallel communication using the various protocols, e.g., UDP, TCP, HTTP proxy, HTTP through port **80** (HTTP **80**), and HTTP through port **8080** (HTTP **8080**). Although only five protocol threads are shown, any number of protocol threads may be initiated by the client, using any conventional and/or suitable protocols. The steps associated with each of threads **794**, **790**, **792**, **796**, **798**, and **788** are discussed herein in connection with FIGS. **8**A–**8**E and **9**.

In FIG. **8**A, the UDP protocol thread is executed. The client inquires in step **716** whether there requires a UDP proxy. If the UDP proxy is required, the user may obtain the name of the UDP proxy from, for example, the system administrator in order to use the UDP proxy to facilitate communication to the proxy (in step **718**). If no UDP proxy is required, the client may directly connect to the server (in step **720**). Thereafter, the client may begin sending a data request (i.e., a control request) to the server in step **722** using the UDP protocol (either through the proxy if a proxy is involved or directly to the server if no proxy is required).

In FIG. **8**B, the TCP protocol thread is executed. If TCP protocol is employed, the client typically directly connects to the server (in step **726**). Thereafter, the client may begin sending a data request (i.e., a control request) to the server using the TCP protocol (step **724**).

In FIG. **8**C, the HTTP protocol thread is executed. The client inquires in step **716** whether there requires a HTTP proxy. If the HTTP proxy is required, the user may obtain the name of the HTTP proxy from, for example, the browser since, as discussed earlier, the data pertaining to the proxy may be kept by the browser. Alternatively, the user may obtain data pertaining to the HTTP proxy from the system administrator in order to use the HTTP proxy to facilitate communication to the server (in step **732**).

If no HTTP proxy is required, the client may directly connect to the server (in step **730**). Thereafter, the client may begin sending a data request (i.e., a control request) to the server in step **734** using the HTTP protocol (either through the proxy if a proxy is involved or directly to the server if no proxy is required).

In FIG. **8**D, the HTTP **80** protocol thread is executed. If HTTP **80** protocol is employed, HTTP data may be exchanged but only through port **80**, which may be for example the port on the client computer through which communication with the network is permitted. Through port **80**, the client typically directly connects to the server (in step **736**). Thereafter, the client may begin sending a data request (i.e., a control request) to the server (step **738**) using the HTTP **80** protocol.

In FIG. **8**E, the HTTP **8080** protocol thread is executed. If HTTP **8080** protocol is employed, HTTP data may be exchanged but only through port **8080**, which may be the port on the client computer for communicating with the network. Through port **8080**, the client typically directly connects to the server (in step **740**). Thereafter, the client may begin sending a data request (i.e., a control request) to the server (step **742**) using the HTTP **8080** protocol. The multiplexing and demultiplexing techniques that may be employed for communication through port **8080**, as well as port **80** of FIG. **8**D, have been discussed earlier and are not repeated here for brevity sake.

FIG. **9** illustrates, in accordance with one embodiment of the present invention, control thread **794** of FIG. **7**. It should be emphasized that FIG. **7** is but one way of implementing the control thread; other techniques of implementing the control thread to facilitate autodetect should be apparent to those skilled in the art in view of this disclosure. In step **746**, the thread determines whether the predefined timeout period has expired. The predefined timeout period may be any predefined duration (such as 7 seconds for example) from the time the data request is sent out to the server (e.g., step **722** of FIG. **8**A). In one embodiment, each protocol thread has its own timeout period whose expiration occurs at the expiration of a predefined duration after the data request using that protocol has been sent out. When all the timeout periods associated with all the protocols have been accounted for, the timeout period for the autodetect technique is deemed expired.

If the timeout has occured, the thread moves to step **754** wherein the most advantageous protocol among the set of protocols received back from the server is selected for communication. As mentioned, the selection of the most advantageous protocol may be performed in accordance with some predefined priority scheme, and data regarding the selected protocol may be saved for future communication sessions between this server and this client.

If no timeout has occurred, the thread proceeds to step **748** to wait for either data from the server or the expiration of the timeout period. If timeout occurs, the thread moves to step **754**, which has been discussed earlier. If data is received from the server, the thread moves to step **750** to ascertain whether the protocol associated with the data received from the server is the predefined "best" protocol, e.g., in accordance with the predefined priority.

If the predefined "best" protocol (e.g., UDP in some real-time data rendering applications) is received, the thread preferably moves to step **754** to terminate the autodetect and to immediately begin using this protocol for data communication instead of waiting of the timeout expiration. Advantageously, the duration of the autodetect sequence may be substantially shorter than the predefined timeout period. In this manner, rapid autodetect of the most suitable protocol and rapid establishment of communication are advantageously facilitated.

If the predefined "best" protocol is not received in step **750**, the thread proceeds to step **752** to add the received protocol to the received set. This received protocol set represents the set of protocols from which the "most advantageous" (relatively speaking) protocol is selected. The most advantageous protocol is ascertained relative to other protocols in the received protocol set irrespective whether it is the predefined "best" protocol in accordance with the predefined priority. As an example of a predefined protocol priority, UDP may be deemed to be best (i.e., the predefined best), followed by TCP, HTTP, then HTTP **80** and HTTP **8080** (the last two may be equal in priority). As mentioned earlier, the most advantageous protocol is selected from the received protocol set preferably upon the expiration of the predefined timeout period.

From step **752**, the thread returns to step **746** to test whether the timeout period has expired. If not, the thread continues along the steps discussed earlier.

Note that since the invention attempts to establish communication between the client application and the server computer in parallel, the time lag between the time the autodetect mechanism begins to execute and the time when the most advantageous protocol is determined is minimal. If communication attempts have been tried in serial, for example, the user would suffer the delay associated with each protocol thread in series, thereby disadvantageously lengthening the time period between communication attempt and successful establishment of communication.

The saving in time is even more dramatic in the event the network is congested or damaged. In some networks, it may take anywhere from 30 to 90 seconds before the client application realizes that an attempt to connect to the server (e.g., step **720**, **726**, **730**, **736**, or **740**) has failed. If each protocol is tried in series, as is done in one embodiment, the delay may, in some cases, reach minutes before the user realizes that the network is unusable and attempts should be made at a later time.

By attempting to establish communication via the multiple protocols in parallel, network-related delays are suffered in parallel. Accordingly, the user does not have to wait for multiple attempts and failures before being able to ascertain that the network is unusable and an attempt to establish communication should be made at a later time. In one embodiment, once the user realizes that all parallel attempts to connect with the network and/or the proxies have failed, there is no need to make the user wait until the expiration of the timeout periods of each thread. In accordance with this embodiment, the user is advised to try again as soon as it is realized that parallel attempts to connect with the server have all failed. In this manner, less of the user's time is needed to establish optimal communication with a network.

While this invention has been described in terms of several preferred embodiments, there are alterations, permutations, and equivalents which fall within the scope of this invention. For example, although the invention has been described with reference with sending out protocol threads in parallel, the automatic protocol detection technique also applies when the protocol threads are sent serially. In this case, while it may take longer to select the most advantageous protocol for selection, the automatic protocol detection technique accomplishes the task without requiring any sophisticated technical knowledge on the part of the user of the client computer. The duration of the autodetect technique, even when serial autodetect is employed, may be shortened by trying the protocols in order of their desirability and ignoring less desirable protocols once a more desirable protocol is obtained. It should also be noted that there are many alternative ways of implementing the methods and apparatuses of the present invention. It is therefore intended that the following appended claims be interpreted as including all such alterations, permutations, and equivalents as fall within the true spirit and scope of the present invention.

What is claimed is:

1. In a computer network, a method for automatically detecting a most advantageous protocol for communication by a client computer, said client computer being configured to be coupled to a server computer via a computer network, the method comprising:

sending, from said client computer to said server computer, a plurality of data requests substantially in parallel, each of said data requests employing a different protocol and a different connection, said data requests being configured to solicit, responsive to said data requests, a set of responses from said server computer, each of said responses employing a protocol associated with a respective one of said data requests;

receiving at said client computer at least a subset of said responses; monitoring said subset of said responses as each response is received from said server computer to select said most advantageous protocol from protocols associated with said subset of said responses, wherein said most advantageous protocol is determined based on a predefined protocol priority that designates a predefined best protocol; and

in response to expiration of a timeout period without receiving a response employed by the predefined best protocol, selecting the most advantageous protocol from the protocols employed by the responses received at the client computer, the selection based at least in part on at least one of a data transfer rate and transmission control characteristics, said timeout period being measured from a transmitting time of a data request.

2. The method of claim **1** wherein said plurality of data requests are sent substantially at the same time.

3. The method of claim **1** wherein said plurality of data requests execute concurrently.

4. The method of claim **1** wherein said client computer, upon receiving a response employing said predefined best protocol, immediately designates said predefined best protocol as said most advantageous protocol.

5. The method of claim 1 wherein at least one data request is sent using a multiplexed HTTP protocol.

6. The method of claim **1** wherein said client computer is configured and arranged to execute an application for rendering real-time data as said real-time data is received from said server computer.

7. The method of claim **1** wherein said computer network comprises the Internet and said predefined best protocol is UDP.

8. The method of claim **7** wherein said real-time data represents one of a video data stream, an audio data stream, and an annotation data stream.

9. In a computer network, a method for automatically detecting a most advantageous protocol for communication by a client computer, said client computer being configured to be coupled to a server computer via a computer network, the method comprising:

sending, from said client computer to said server computer, a plurality of data requests substantially in parallel, each of said data requests employing a different protocol and a different connection;

receiving at least a subset of said data requests at said server computer;

sending a set of responses from said server computer to said client computer, said set of responses being responsive to said subset of said data requests, each of said responses employing a protocol associated with a respective one of said subset of said data requests;

receiving at said client computer at least a subset of said responses; and

selecting, for said communication between said client computer and said server computer, said most advantageous protocol from protocols associated with said subset of said responses, wherein said most advantageous protocol is determined based on a predefined protocol priority that designates a predefined best protocol if a response employing the predefined best protocol is received at the client computer, and wherein, if a timeout period expires without the client computer receiving a response employing the predefined best protocol, the most advantageous protocol is selected from the protocols employed by the responses received at the client computer based at least in part on at least one of a data transfer rate and transmission control characteristics, said timeout period being measured from a transmitting time of a data request.

10. The method of claim **9** wherein selecting the most advantageous protocol comprises:

monitoring, employing said client computer, said subset of said responses as each one of said subset of said responses is received at said client computer from said server computer for a response employing said predefined best protocol; and

designating said predefined best protocol said most advantageous protocol, thereby immediately permitting said client computer to employ said predefined best protocol for communication with said server computer without further receiving additional responses from said server computer.

11. The method of claim **9** wherein said computer network comprises the Internet and said predefined best protocol is UDP.

12. The method of claim **9** wherein said client computer is configured and arranged to execute an application for rendering real-time data as said real-time data is received from said server computer.

13. The method of claim **12** wherein said real-time data represents one of a video data stream, an audio data stream, and an annotation data stream.

14. The method of claim **13** wherein said data requests employ at least one of a UDP, TCP, HTTP proxy, HTTP **80**, and HTTP **8080** protocols.

15. The method of claim **14** wherein said HTTP **80** protocol is a protocol for permitting multiple HTTP data streams to be transmitted in a multiplexed manner through port **80**, said multiple HTTP data streams representing said video data stream and said audio data stream.

16. The method of claim **14** wherein said HTTP **8080** protocol is a protocol for permitting multiple HTTP data streams to be transmitted in a multiplexed manner through port **8080**, said multiple HTTP data streams representing said video data stream and said audio data stream.

17. A computer-readable medium containing computer-readable instructions for automatically detecting a most advantageous protocol for communication by a client computer, said client computer being configured for coupling to a server computer via a computer network, said computer-readable instructions comprise:

computer-readable instructions for sending in a substantially parallel manner, from said client computer to said server computer, a plurality of data requests, each of said data requests employing a different protocol and a different connection, said data requests being configured to solicit, responsive to said data requests, a set of responses from said server computer, each of said responses employing a protocol associated with a respective one of said data requests;

computer-readable instructions for receiving at said client computer at least a subset of said responses;

computer-readable instructions for monitoring said subset of said responses as each response is received from said server computer to select said most advantageous protocol from protocols associated with said subset of said responses, wherein said most advantageous protocol is determined based on a predefined protocol priority that designates a predefined best protocol; and

computer-readable instructions for, in response to expiration of a timeout period without receiving a response employing the predefined best protocol, selecting the most advantageous protocol from the protocols employed by the responses received at the client computer, the selection based at least in part on at least one of a data transfer rate and transmission control characteristics, said timeout period being measured from a transmitting time of a data request.

18. The computer-readable medium of claim **17** wherein said client computer, upon receiving a response employing said predefined best protocol, immediately designates said predefined best protocol said most advantageous protocol.

19. The computer-readable medium of claim **17** wherein at least one data request is sent using a multiplexed HTTP protocol.

20. The computer-readable medium of claim **17** wherein said client computer is configured and arranged to execute an application for rendering real-time data as said real-time data is received from said server computer.

21. The computer-readable medium of claim **17** wherein said computer network comprises the Internet and said predefined best protocol is UDP.

22. The computer readable medium of claim **21** wherein said real-time data represents one of a video data stream, an audio data stream, and an annotation data stream.

23. A computer-readable medium containing computer-readable instructions for automatically detecting a most advantageous protocol for communication by a client computer, said client computer being, configured for coupling to a server computer via a computer network, said computer-readable instructions comprise:

computer-readable instructions for sending from said client computer to said server computer, a plurality of data requests substantially in parallel, each of said data requests employing a different protocol and a different connection;

computer-readable instructions for receiving, at least a subset of said data requests at said server computer;

computer-readable instructions for sending a set of responses from said server computer to said client computer, said set of responses being responsive to said subset of said data requests, each of said responses employing a protocol associated with a respective one of said subset of said data requests;

computer-readable instructions for receiving at said client computer at least a subset of said responses; and

computer-readable instructions for selecting, for said communication between said client computer and said server computer, said most advantageous protocol from protocols associated with said subset of said responses, wherein said most advantageous protocol is determined based on a predefined protocol priority that designates a predefined best protocol if a response employing the predefined best protocol is received at the client computer, and wherein, if a timeout period expires without the client computer receiving a response employing the predefined best protocol, the most advantageous protocol is selected from the protocols employed by the responses received at the client computer based at least in part on at least one of a data transfer rate and transmission control characteristics, said timeout period being measured from a transmitting time of a data request.

* * * * *

(B) Evidence for Claim 4 – Relied Upon

The following item (1) listed below is hereby entered as evidence relied upon by the Examiner as to grounds of rejection for claim 4, to be reviewed on appeal. Also listed for each item is where said evidence was entered into the record by the Examiner.

(1) Copy of US Patent Number 5,852,717 ("Bhide et al"). This evidence was entered into the record by the Examiner on page 5 at 8 of the Office Action mailed 08/10/2006.

Copies of all References follows.

//

US005852717A

United States Patent [19]

Bhide et al.

[11] **Patent Number: 5,852,717**

[45] **Date of Patent: Dec. 22, 1998**

[54] **PERFORMANCE OPTIMIZATIONS FOR COMPUTER NETWORKS UTILIZING HTTP**

[75] Inventors: **Chandrashekhar W. Bhide**, Sunnyvale; **Jagdeep Singh**; **Don Oestreicher**, both of Cupertino, all of Calif.

[73] Assignee: **Shiva Corporation**, Bedford, Mass.

[21] Appl. No.: **752,500**

[22] Filed: **Nov. 20, 1996**

[51] **Int. Cl.**^{6} **G06F 17/00**

[52] **U.S. Cl.** **395/200.33**; 395/200.49; 395/200.59

[58] **Field of Search** 395/200.32, 200.33, 395/200.58, 200.59, 200.49, 200.5; 364/551.01

[56] **References Cited**

U.S. PATENT DOCUMENTS

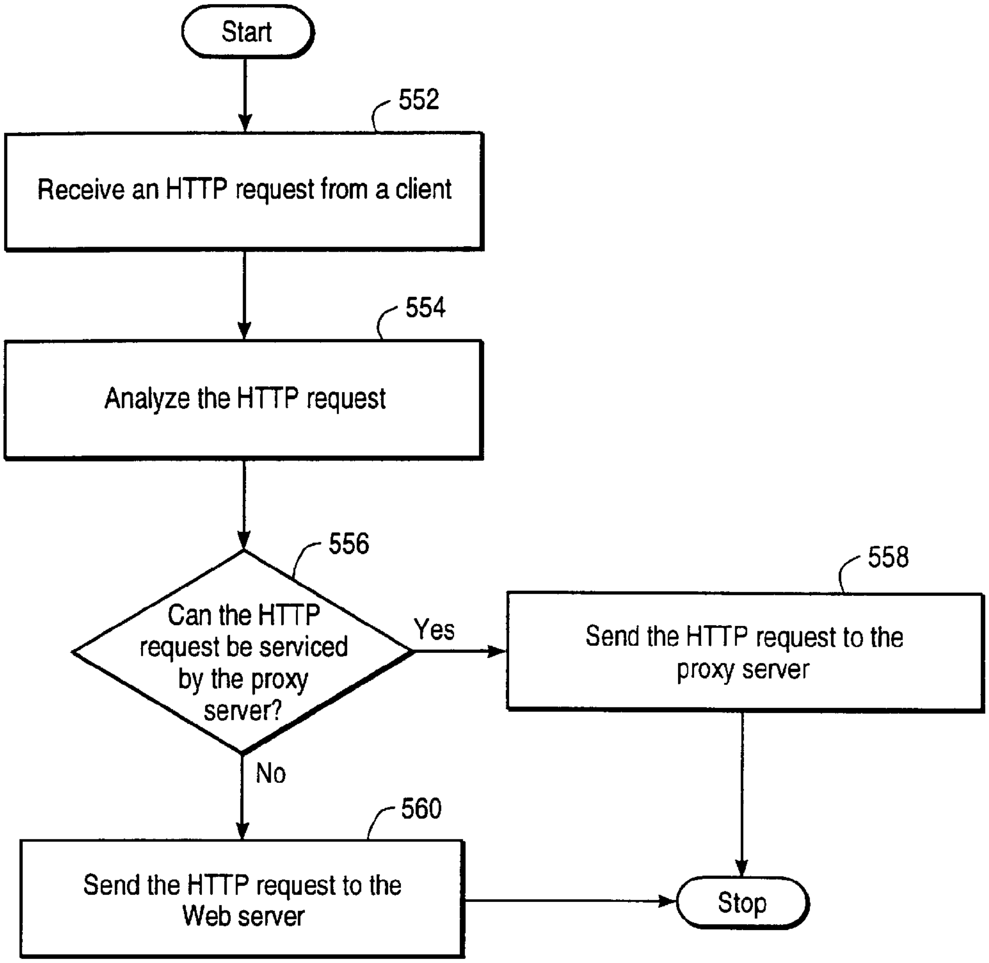
4,891,785 1/1990 Donohoo 395/200.32
5,339,435 8/1994 Lubkin et al. 395/200.5
5,710,918 1/1998 Lagarde et al. 395/200.32

Primary Examiner—Ellis B. Ramirez
Attorney, Agent, or Firm—Townsend and Townsend and Crew LLP

[57] **ABSTRACT**

Systems and methods of increasing the performance of computer networks, especially networks connecting users to the Web, are provided. Performance is increased by reducing the latency the client experiences between sending a request to the server and receiving a response. A connection cache may be maintained by an agent on the network access equipment to more quickly respond to request for network connections to the server. Additionally, the agent may maintain a cache of information to more quickly respond to requests to get an object if it has been modified. These enhancements and other described herein may be implemented singly or in conjunction to reduce the latency involved in sending the requests to the server by saving round-trip times between computer network components.

48 Claims, 15 Drawing Sheets


Start
552
Receive an HTTP request from a client
554
Analyze the HTTP request
556
Can the HTTP request be serviced by the proxy server?
Yes
558
Send the HTTP request to the proxy server
No
560
Send the HTTP request to the Web server
Stop

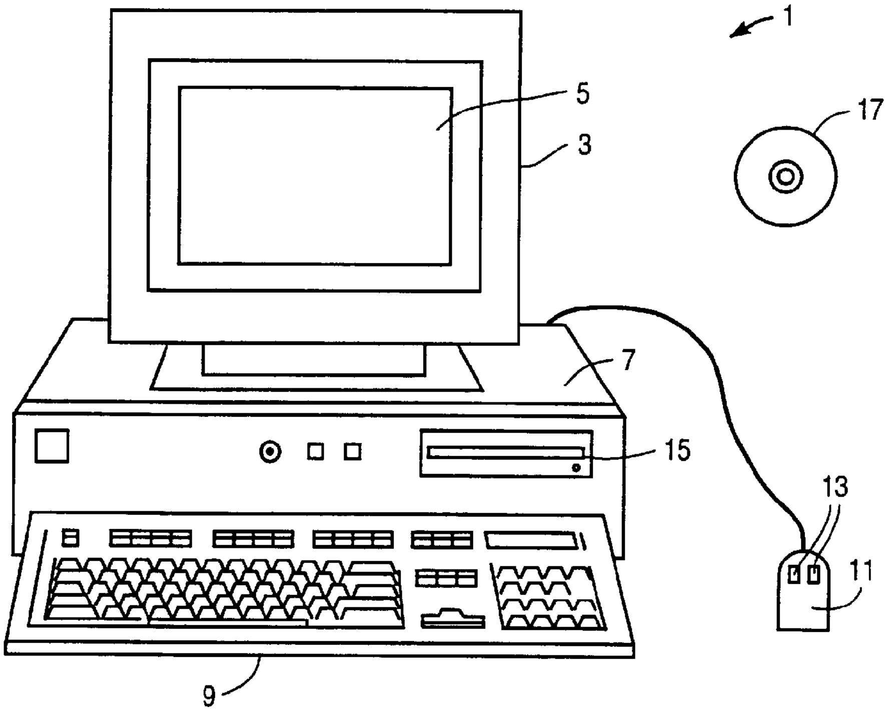

1
5
3
17
7
15
13
11
9

FIG. 1

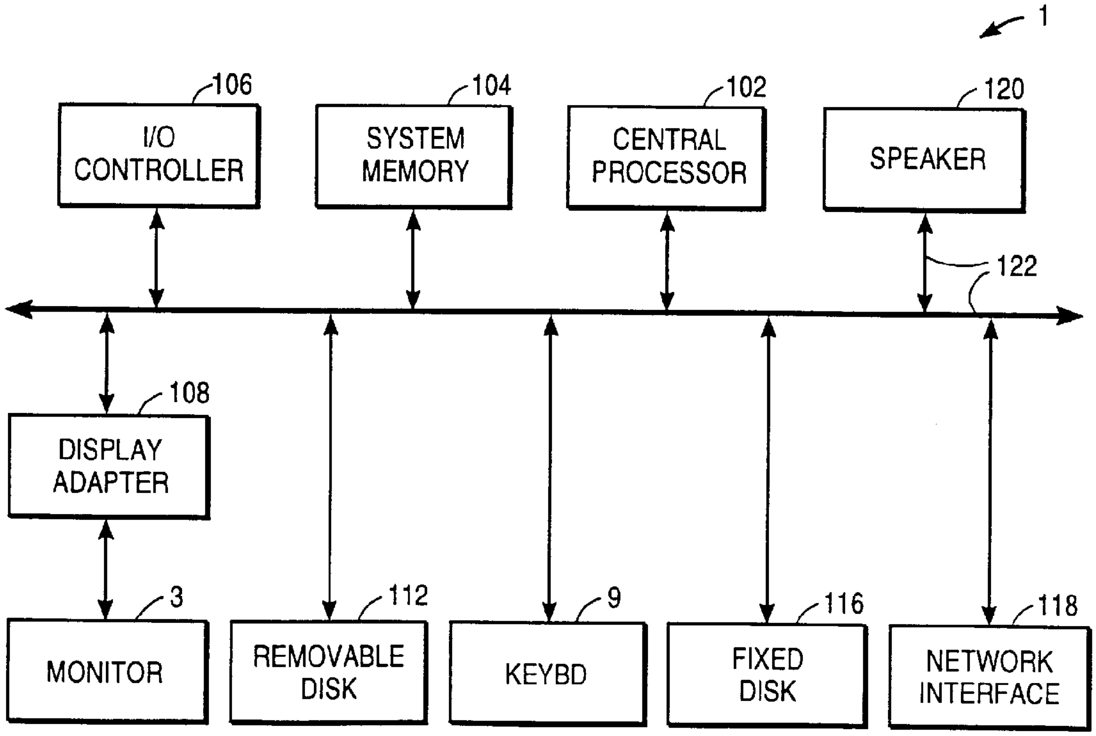

1
106
I/O CONTROLLER
104
SYSTEM MEMORY
102
CENTRAL PROCESSOR
120
SPEAKER
122
108
DISPLAY ADAPTER
3
MONITOR
112
REMOVABLE DISK
9
KEYBD
116
FIXED DISK
118
NETWORK INTERFACE

FIG. 2

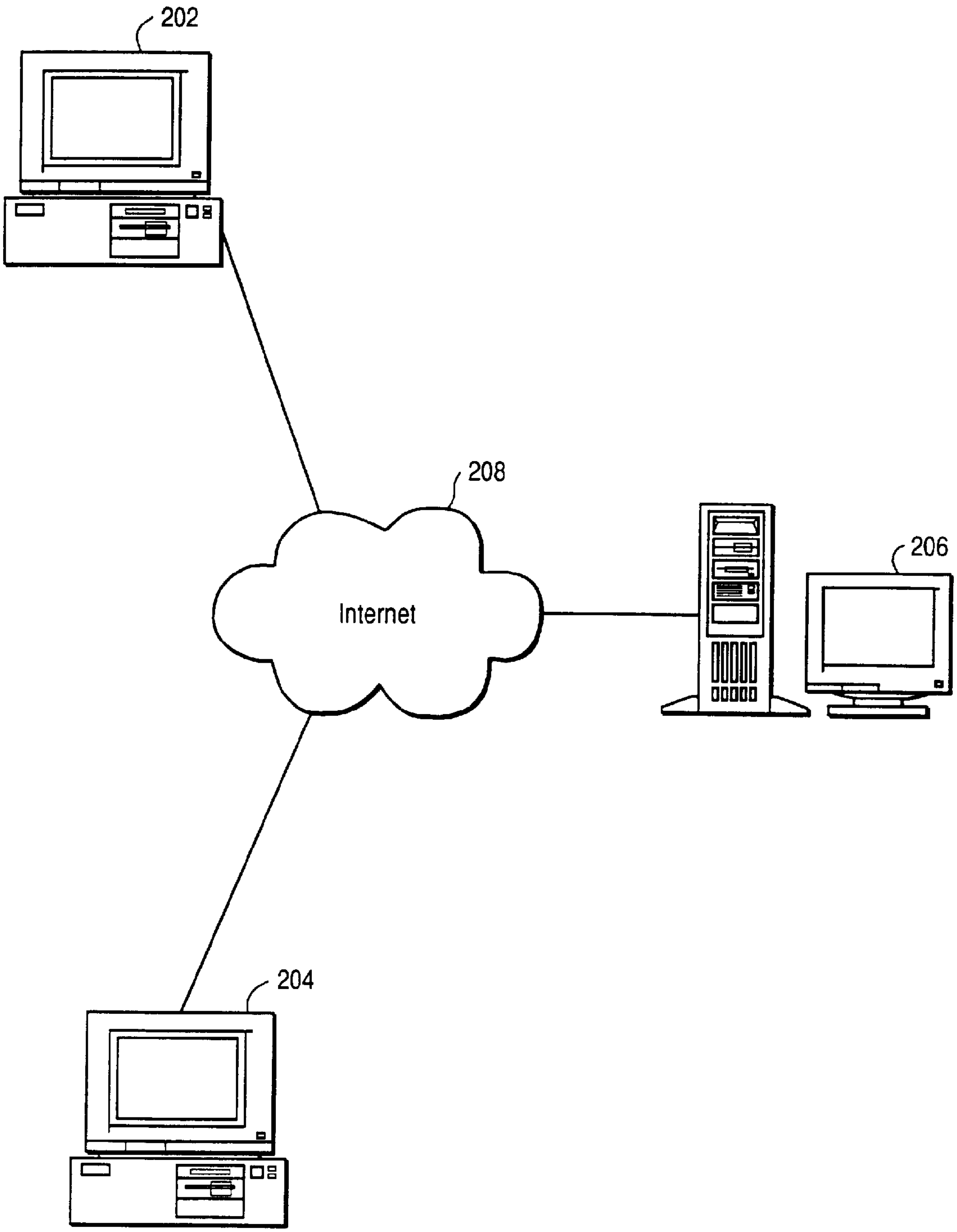


FIG. 3

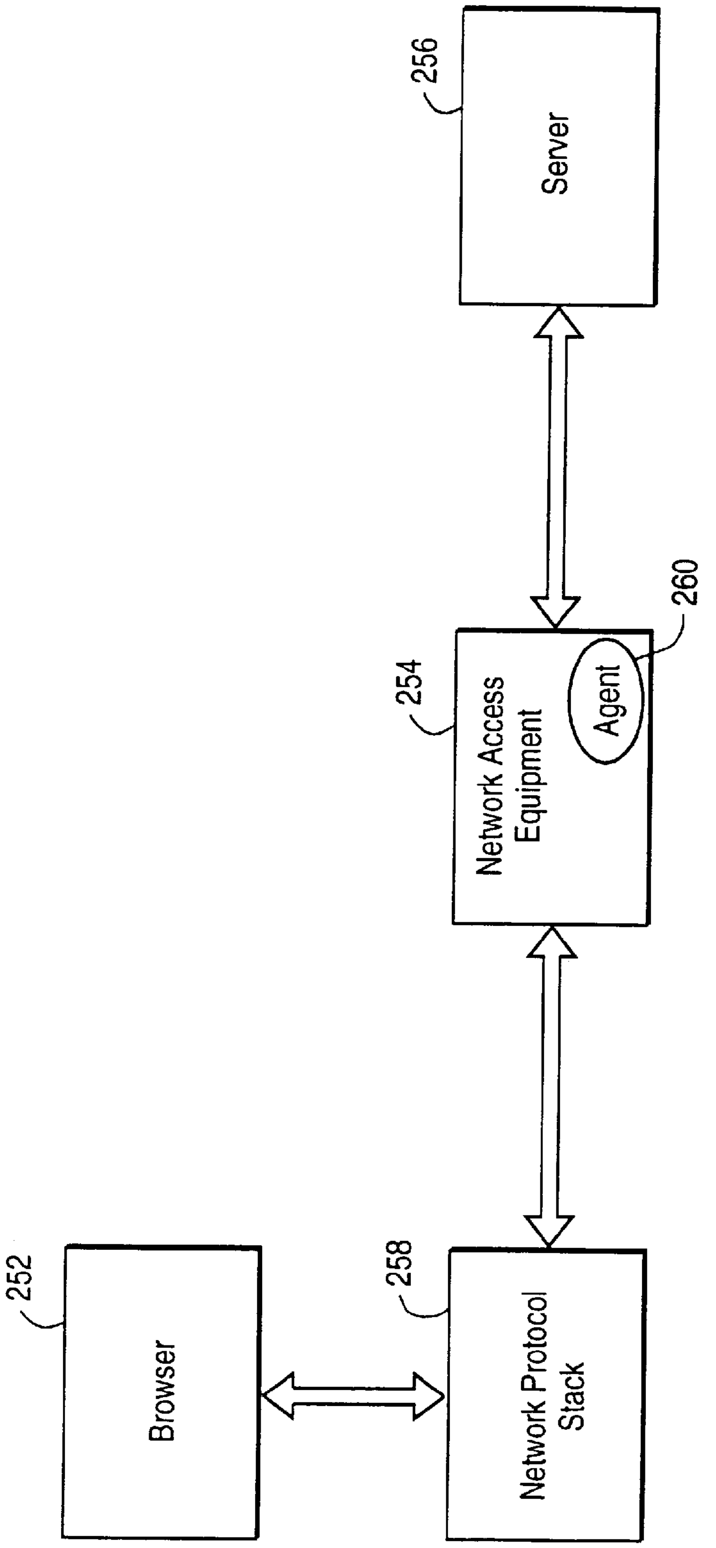

Browser
252
Network Protocol Stack
258
Network Access Equipment
254
Agent
260
Server
256

FIG. 4

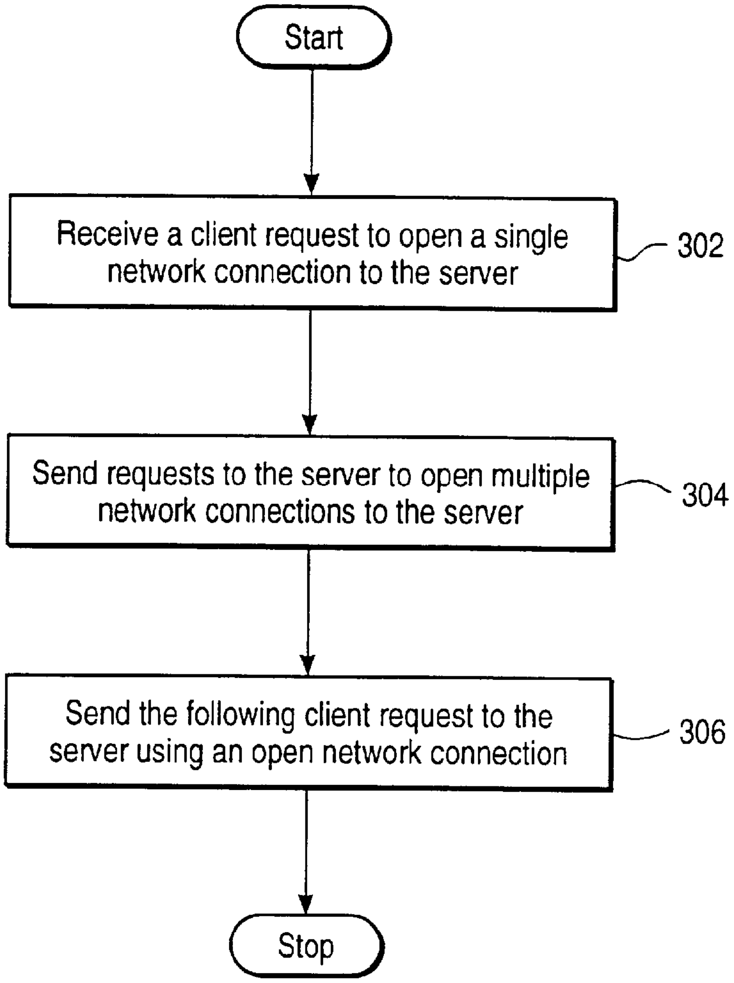

Start
Receive a client request to open a single network connection to the server
302
Send requests to the server to open multiple network connections to the server
304
Send the following client request to the server using an open network connection
306
Stop

FIG. 5

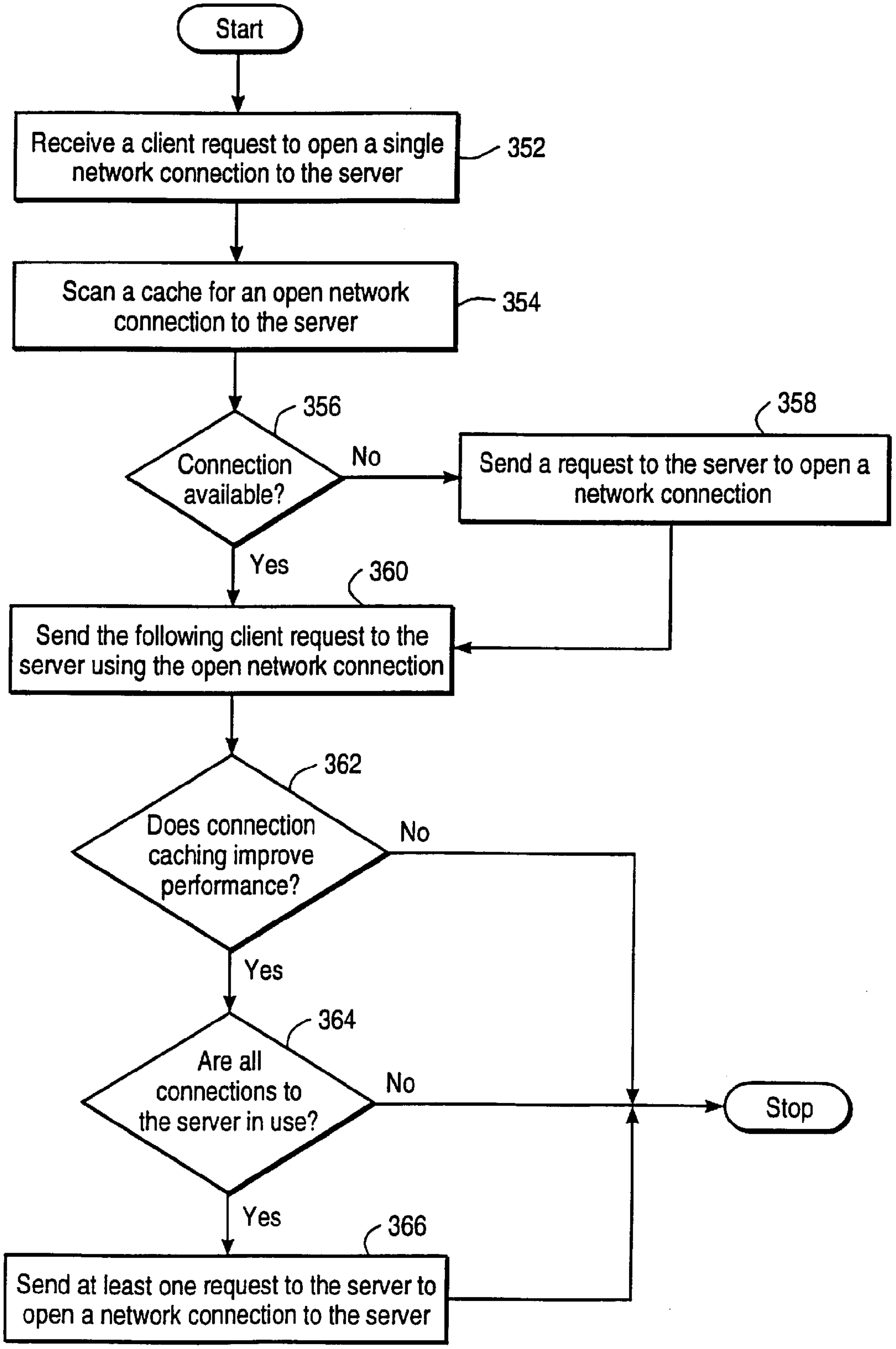

Start
Receive a client request to open a single network connection to the server
352
Scan a cache for an open network connection to the server
354
356
Connection available?
No
Send a request to the server to open a network connection
358
Yes
360
Send the following client request to the server using the open network connection
362
Does connection caching improve performance?
No
Yes
364
Are all connections to the server in use?
No
Stop
Yes
366
Send at least one request to the server to open a network connection to the server

FIG. 6

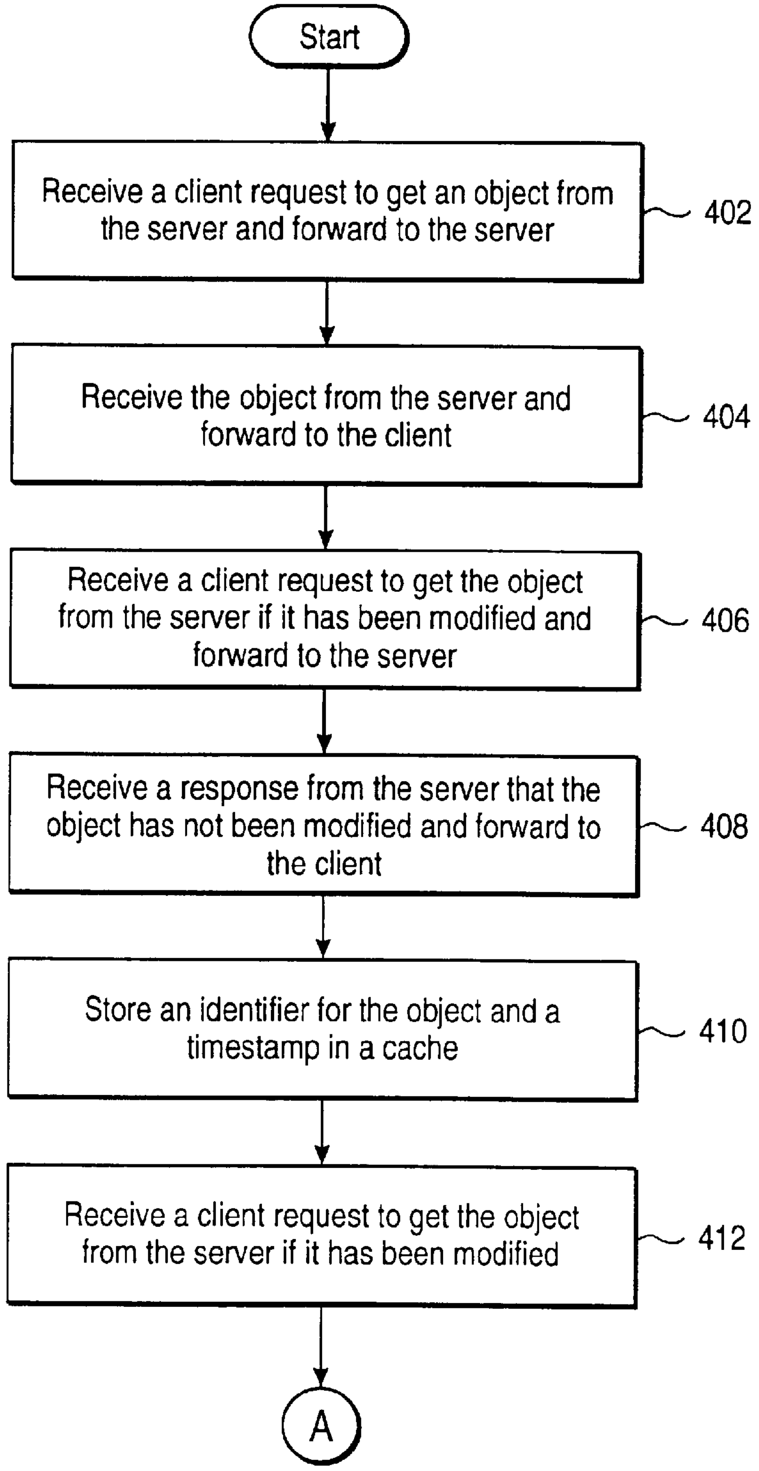

Start
Receive a client request to get an object from the server and forward to the server
402
Receive the object from the server and forward to the client
404
Receive a client request to get the object from the server if it has been modified and forward to the server
406
Receive a response from the server that the object has not been modified and forward to the client
408
Store an identifier for the object and a timestamp in a cache
410
Receive a client request to get the object from the server if it has been modified
412
A

FIG. 7A

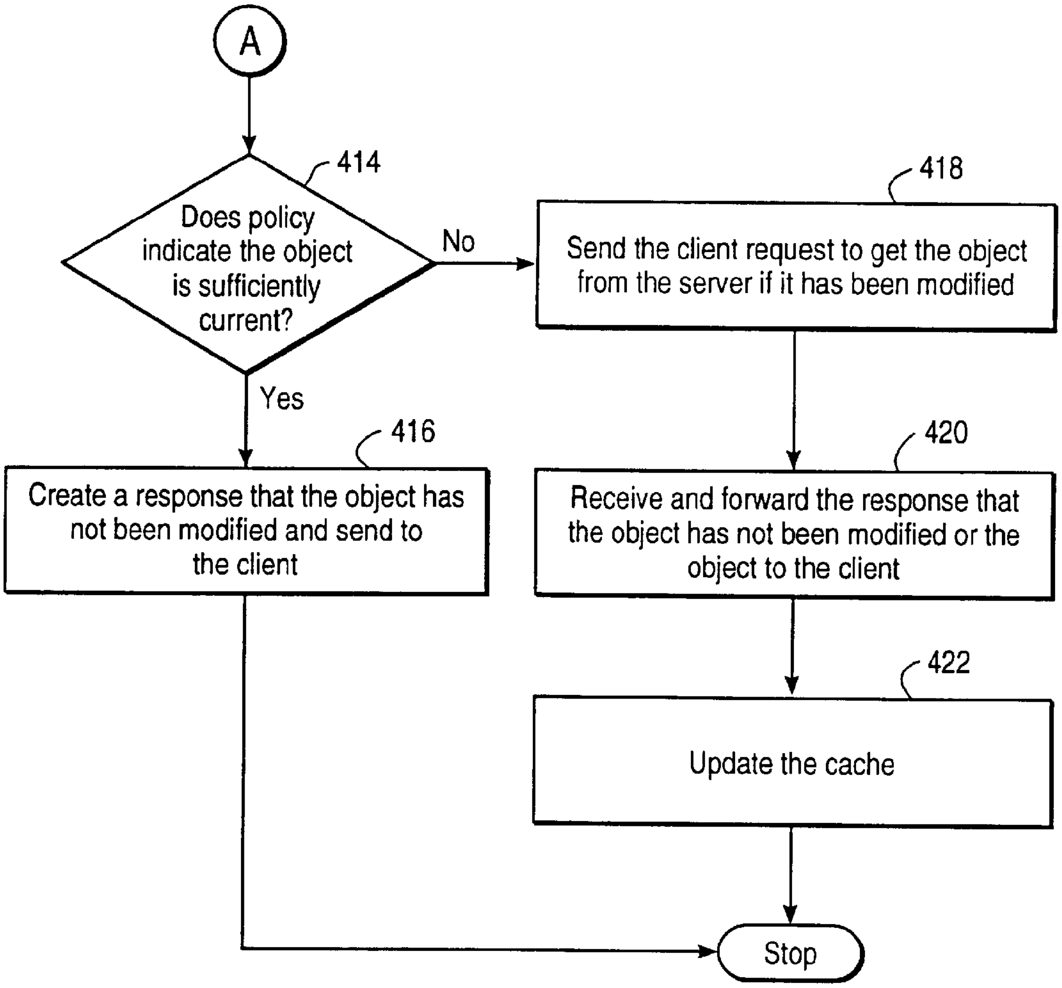

A
414
Does policy indicate the object is sufficiently current?
No
418
Send the client request to get the object from the server if it has been modified
Yes
416
Create a response that the object has not been modified and send to the client
420
Receive and forward the response that the object has not been modified or the object to the client
422
Update the cache
Stop

FIG. 7B

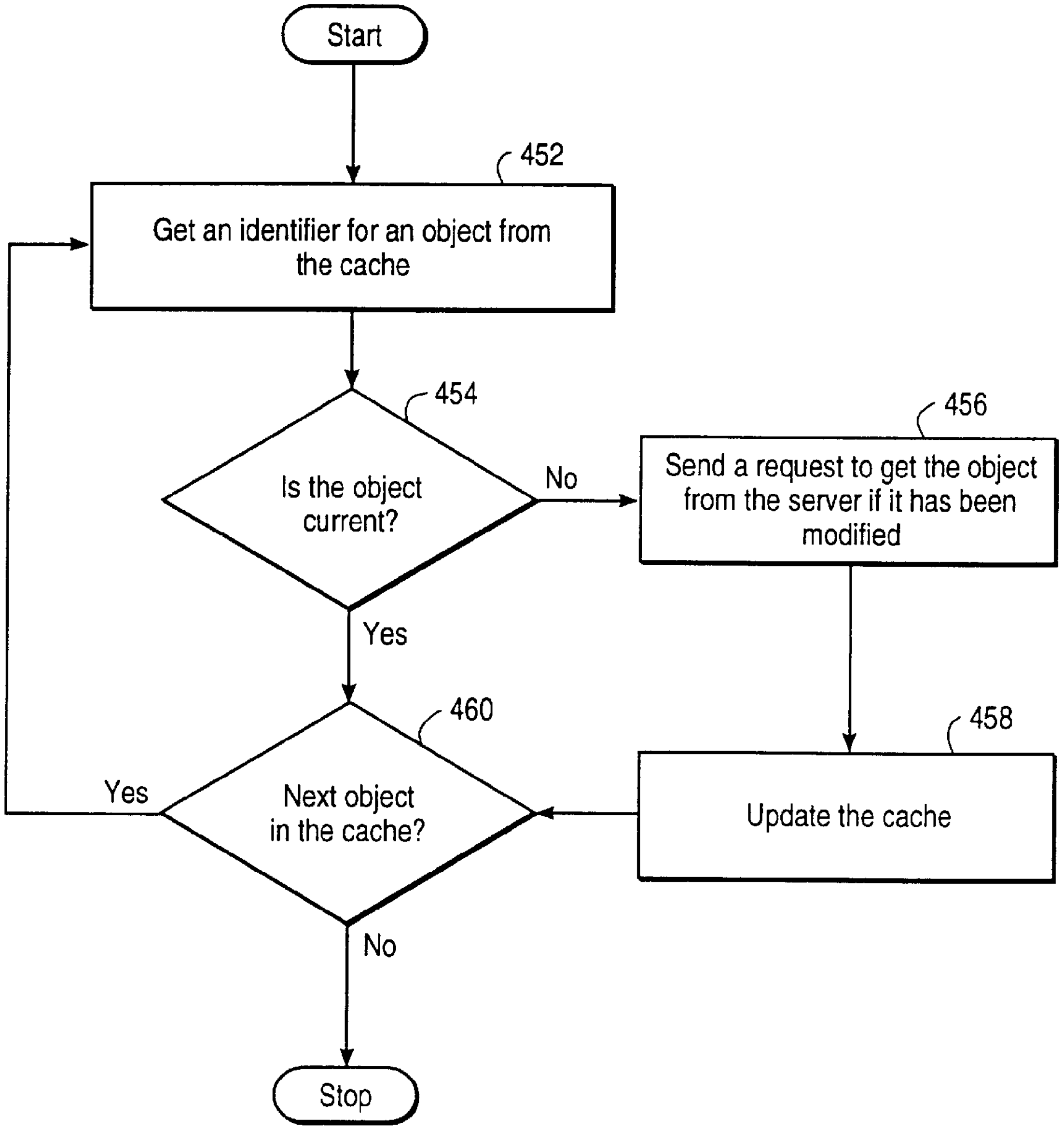

Start
452
Get an identifier for an object from the cache
454
Is the object current?
No
456
Send a request to get the object from the server if it has been modified
Yes
460
458
Yes
Next object in the cache?
Update the cache
No
Stop

FIG. 8

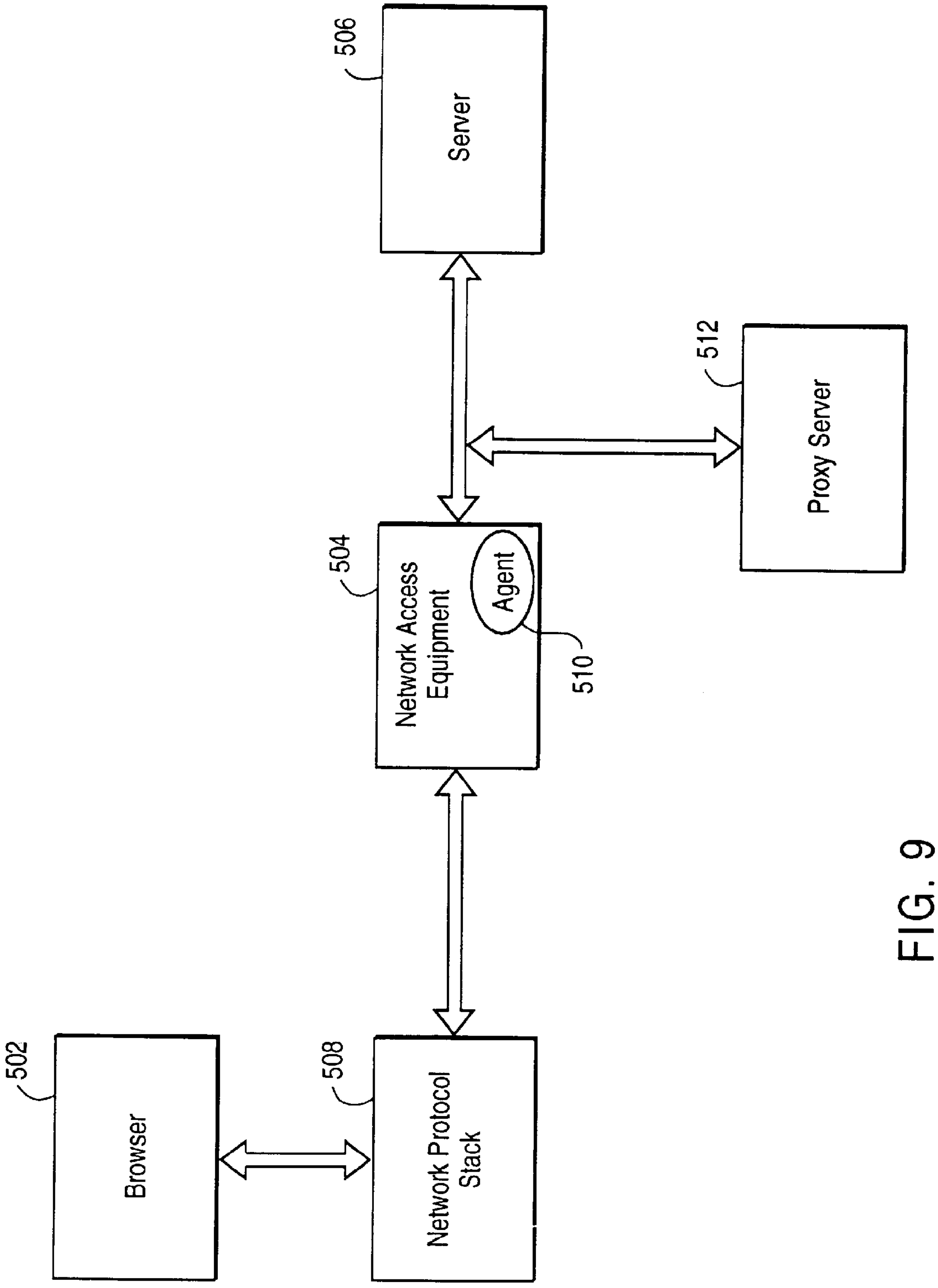

506
Server
512
Proxy Server
504
Network Access Equipment
Agent
510
502
Browser
508
Network Protocol Stack

FIG. 9

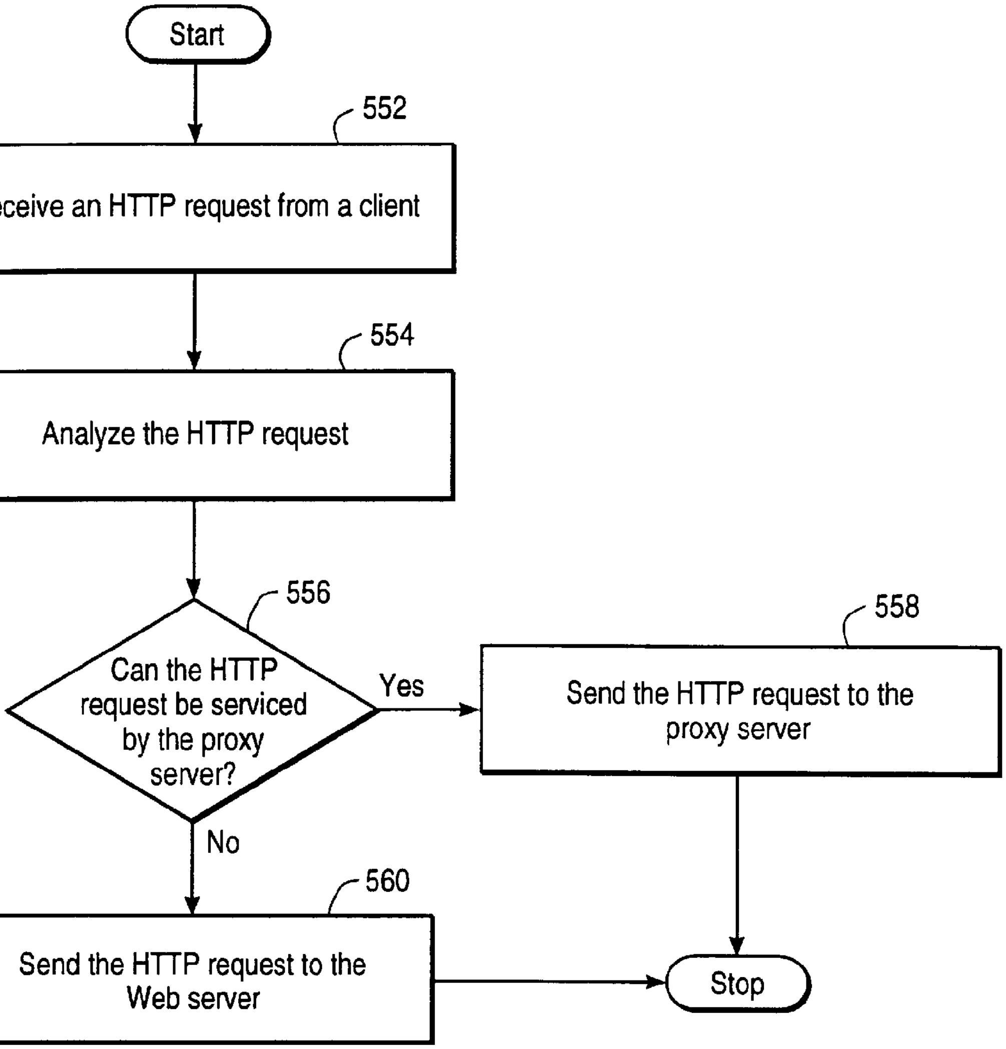

Start
552
Receive an HTTP request from a client
554
Analyze the HTTP request
556
Can the HTTP request be serviced by the proxy server?
Yes
558
Send the HTTP request to the proxy server
No
560
Send the HTTP request to the Web server
Stop

FIG. 10

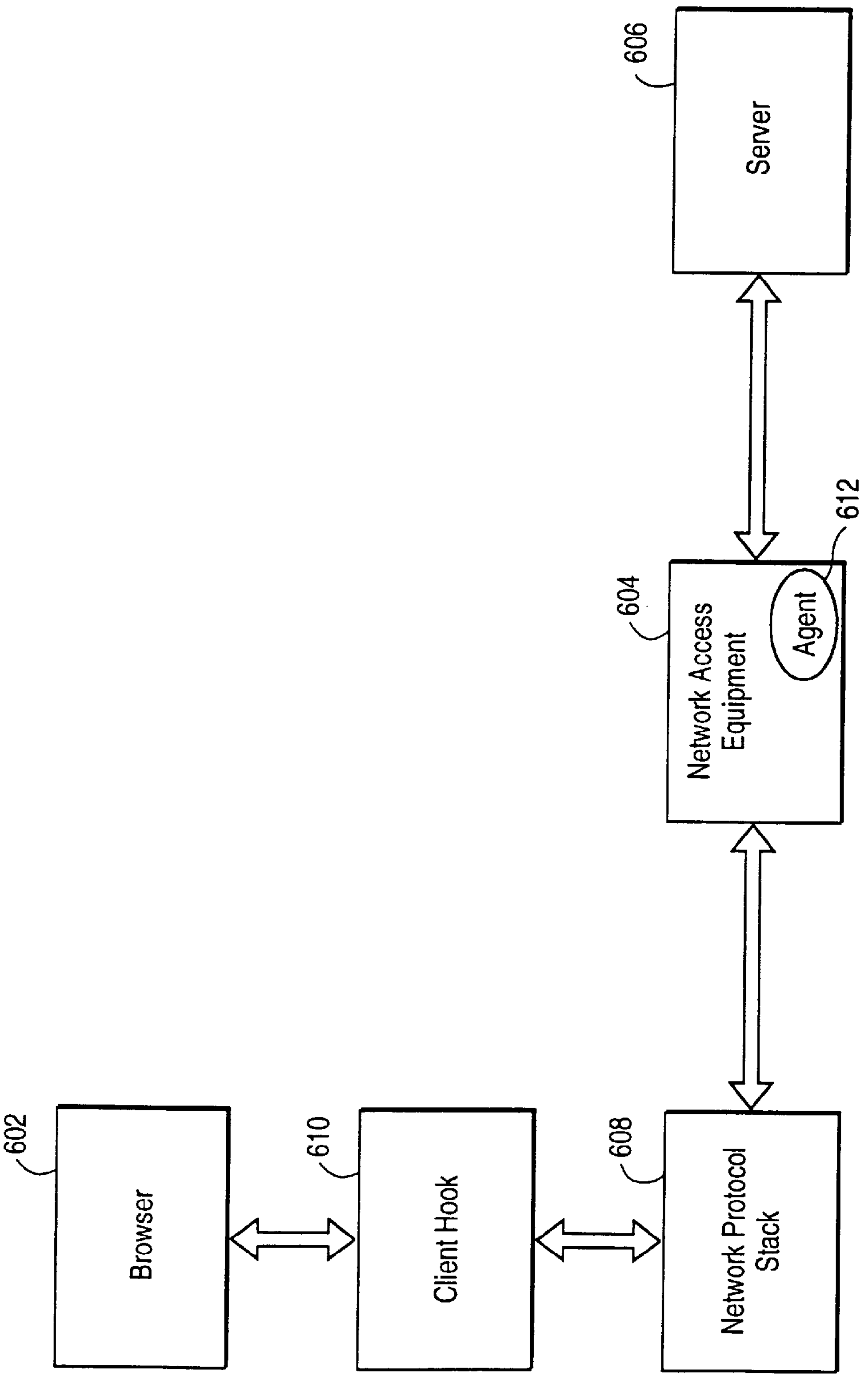

606
Server
604
Network Access Equipment
Agent
612
602
Browser
610
Client Hook
608
Network Protocol Stack

FIG. 11

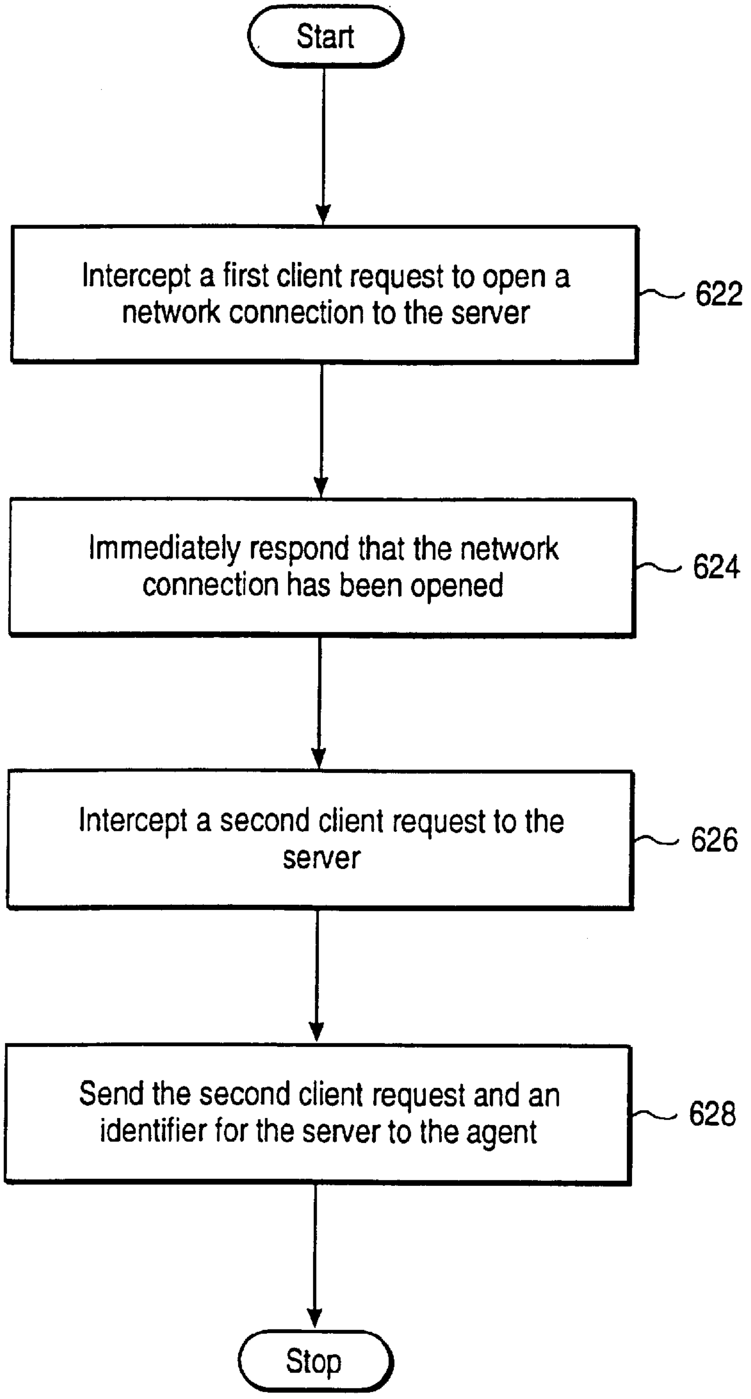

Start
Intercept a first client request to open a network connection to the server
622
Immediately respond that the network connection has been opened
624
Intercept a second client request to the server
626
Send the second client request and an identifier for the server to the agent
628
Stop

FIG. 12

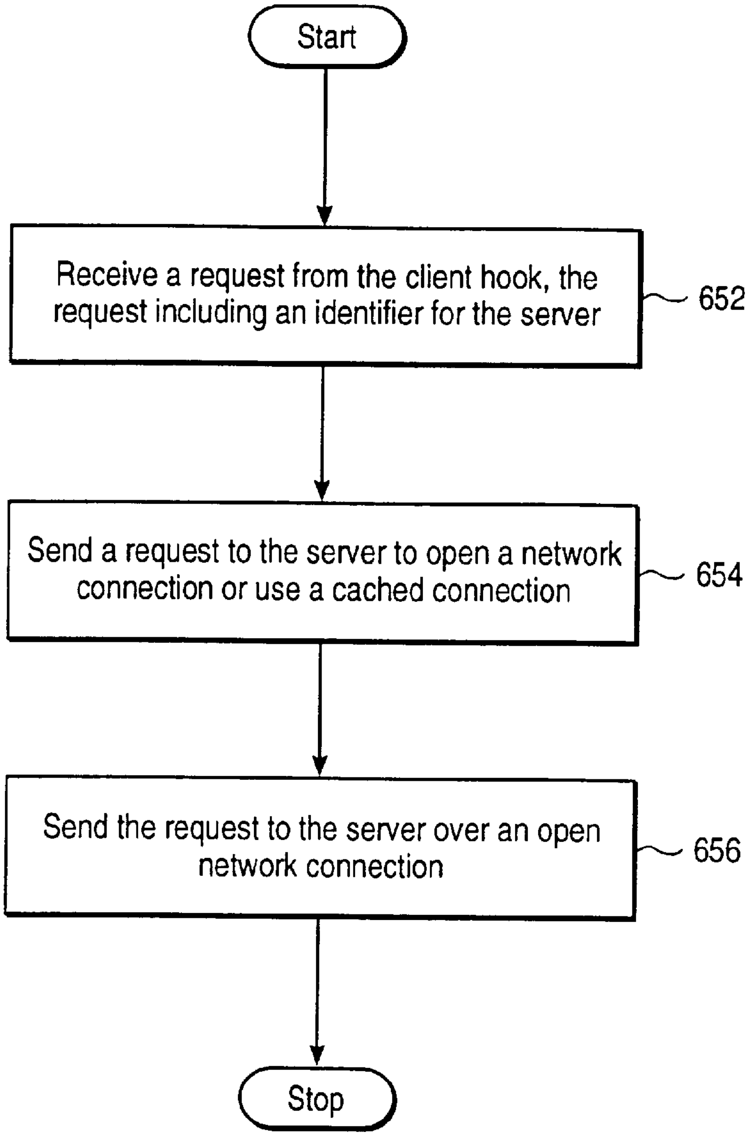

Start
Receive a request from the client hook, the request including an identifier for the server
652
Send a request to the server to open a network connection or use a cached connection
654
Send the request to the server over an open network connection
656
Stop

FIG. 13

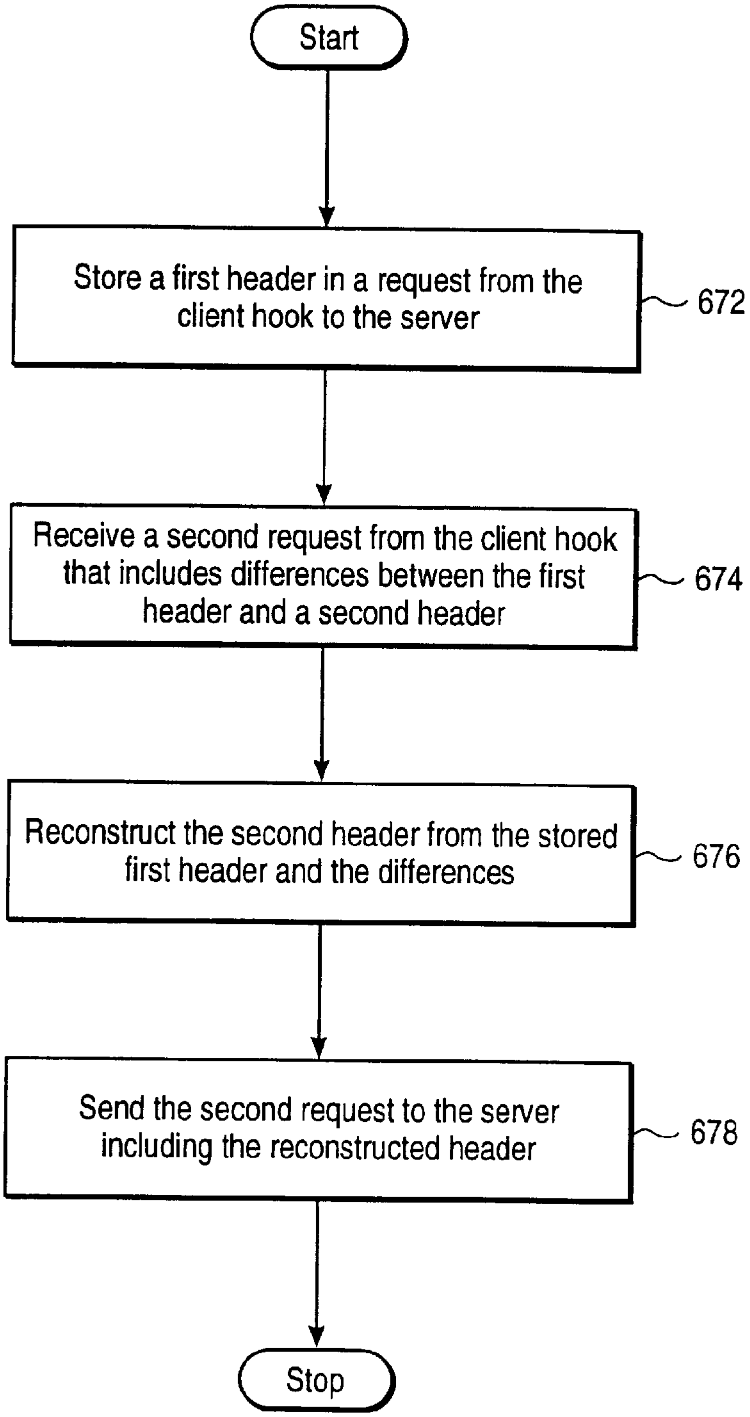

Start
Store a first header in a request from the client hook to the server
672
Receive a second request from the client hook that includes differences between the first header and a second header
674
Reconstruct the second header from the stored first header and the differences
676
Send the second request to the server including the reconstructed header
678
Stop

FIG. 14

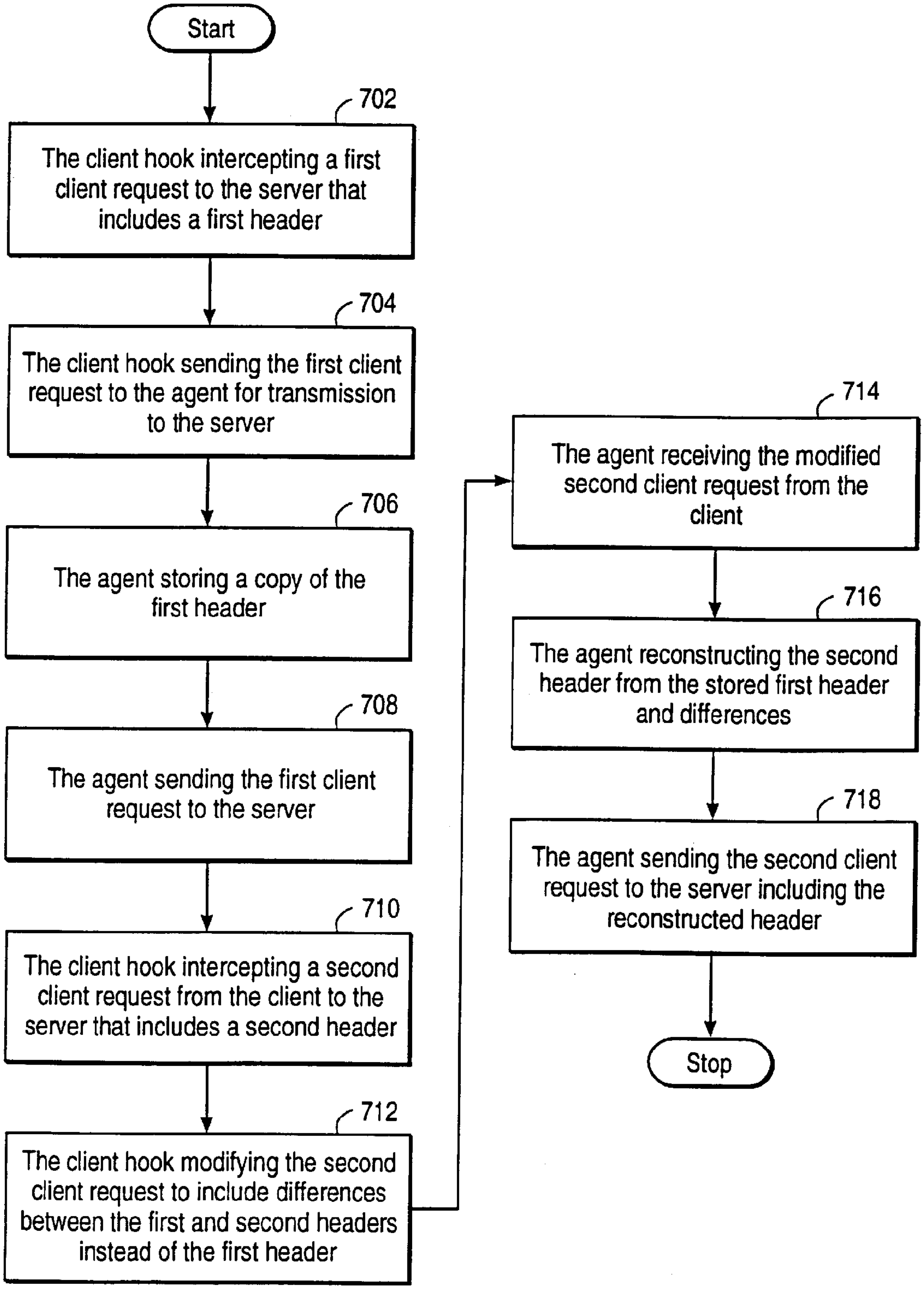

Start
702
The client hook intercepting a first client request to the server that includes a first header
704
The client hook sending the first client request to the agent for transmission to the server
706
The agent storing a copy of the first header
708
The agent sending the first client request to the server
710
The client hook intercepting a second client request from the client to the server that includes a second header
712
The client hook modifying the second client request to include differences between the first and second headers instead of the first header
714
The agent receiving the modified second client request from the client
716
The agent reconstructing the second header from the stored first header and differences
718
The agent sending the second client request to the server including the reconstructed header
Stop

FIG. 15

PERFORMANCE OPTIMIZATIONS FOR COMPUTER NETWORKS UTILIZING HTTP

BACKGROUND OF THE INVENTION

The present invention is related to increasing performance of networked computer systems and, more particularly, increasing the performance of computer systems accessing the World Wide Web ("Web") on the Internet.

The Internet is a network which provides avenues for worldwide communication of information, ideas and messages. Although the Internet has been utilized by academia for decades, recently there has been almost an explosion of interest in the Internet and the information residing thereon. The Web accounts for a significant part of the growth in the popularity of the Internet, perhaps because of the user-friendly graphical user interfaces ("GUIs") of browsers that are readily available for accessing the Web.

The World Wide Web makes hypertext documents available to users over the Internet. A hypertext document does not present information linearly like a book, but instead provides the reader with links or pointers to other locations so that the user may jump from one location to another. The hypertext documents on the Web are accessed through the client/server protocol of Hypertext Transport Protocol ("HTTP").

The Internet utilizes the Transmission Control Protocol/Internet Protocol ("TCP/IP") to network very diverse and dissimilar systems. In Windows 3.x environments, the browser typically utilizes a dynamic link library WINSOCK.DLL to communicate with the TCP/IP-based Internet. Although the hardware backbone of the Internet is a series of high-speed communications links between educational, research, government, and commercial mainframe computer systems, a great number of the users that access the Web utilize a browser that is connected to the Internet through a relatively slow or weak link (e.g., a 28.8K modem over an analog phone line) to network access equipment networked to the Internet.

The network access equipment typically has a fast connection to the Internet (e.g., a T-1 connection at 1.54 MB). Network access equipment may be a remote access server for allowing remote users to connect to intranet and Internet resources. Such a remote access server, the LanRover™ Access Switch remote access server, is available from Shiva Corporation, Bedford, Mass. Other types of network access equipment are utilized by Internet Service Providers ("ISPs") to provide Internet access to customers. Thus, the network access equipment is networked between the computer running the browser and the Web server providing what is called the Point of Presence ("POP") for the user.

Network performance in general is hampered because the network link between users and their POP commonly has a significantly lower bandwidth than the network link between the POP and the Web server. Additionally, there is a significant amount of latency in conventional networks while the client waits for a response from the Web server. Accordingly, there is a need for systems and methods for increasing the performance of the computer networks, preferably without requiring modification of existing browsers.

SUMMARY OF THE INVENTION

The present invention provides systems and methods of increasing the performance of computer networks, especially networks connecting users to the Web. Performance may be increased by reducing the latency the client experiences between sending a request to the server and receiving a response. A connection cache may be maintained by an agent on the network access equipment to more quickly respond to request for network connections to the server. Additionally, the agent may maintain a cache of information to more quickly respond to requests to get an object if it has been modified. These enhancements may be implemented singly or in conjunction to reduce the latency involved in sending the respective requests to the server by saving round-trip times between the agent and the server. The invention complements caching provided by browsers and other components (e.g., proxy servers).

Performance may also be increased by the network access equipment sending an HTTP request to either the Web server or a proxy server based on an analysis of the HTTP request. The Web browsers may then utilize a proxy server transparently, without specifically sending requests to the proxy server.

Additionally, performance may be increased by effectively increasing the effectively bandwidth of the weak link between the client and the network access equipment. A client hook intercepts client requests to the server and modifies the client requests to increase performance. The modified requests are then sent to the agent which reconstructs the client requests from the modified requests and sends the client requests to the server. For example, multiple client requests may be combined into a single modified requests or individual client requests may be intelligently compressed for more efficient utilization of the weak link.

In one embodiment, the present invention provides a method executed by an agent in a computer network between clients and a server for increasing performance between the clients and the server, the method comprising the steps of: receiving a first request from a client to open a single network connection to the server; sending a plurality of requests to the server to open a plurality of network connections to the server; receiving a second request from the client; and sending the second request to the server using one of the plurality of network connections. Accordingly, the plurality of network connections to the server are opened in response to the first request from the client to open a single network connection.

In another embodiment, the present invention provides a method executed by an agent in a computer network between clients and a server for increasing performance between the clients and the server, the method comprising the steps of: receiving a first request from a client to get an object from the server if the object has been modified after a specific timestamp; sending the first request to the server; receiving a first response from the server that the object has not been modified after the specific timestamp; sending the first response to the client; storing an identifier for the object and a timestamp in a cache; receiving a second request from the client to get the object from the server if the object has been modified after the specific timestamp; and if the timestamp stored in the cache is within a predetermined amount of time from the current time, sending a second response to the client that the object has not been modified after the specific timestamp without sending the second request to the server.

In another embodiment, the present invention provides a method executed by an agent in a computer network between a client and a Web and proxy servers for increasing performance between the client and the Web server, comprising the steps of: receiving an HTTP request from a client; and sending the HTTP request to either the Web server or the proxy server depending on the HTTP request, the proxy server storing information available on the Web server. Accordingly, the client does not need to be modified or configured to utilize the proxy server.

In another embodiment, the present invention provides a method for increasing performance between a client on a client computer and a server utilizing a client hook on the client computer and an agent between the client computer and the server, comprising the steps of: the client hook intercepting requests from the client to the server; the client hook modifying the requests from the client; the client hook sending the modified requests to the agent; the agent reconstructing the requests from the client according to the modified requests; and the agent sending the requests from the client to the server. The client hook may intercept requests from the client to open a network connection to the server and immediately respond so that the client does not have to wait for a response that a network connection is open. The agent may open the network connection when required or store a cache of open network connections to the server. Also, the client hook may intercept requests from the client to compress information into changes from information in a previous request. The agent has the previous information stored and reconstructs the hew information from the changes. Thus, the communication between the client hook and the agent increases performance of communication between the client and the server.

A feature of the present invention is that performance is increased without necessitating modification of the client or server. As no modifications of a Web browser is required, the enhancements may be implemented to transparently increase the performance of the browser, regardless of the browser that is utilized. Other features and advantages of the present invention will become apparent upon a perusal of the remaining portions of the specification and drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates an example of a computer system used to execute software of an embodiment of the present invention;

FIG. **2** shows a system block diagram of a typical computer system used to execute software of an embodiment of the present invention;

FIG. **3** shows a diagram of multiple computers networked over the Internet;

FIG. **4** is a block diagram of a browser connected to a Web server through network access equipment including an agent;

FIG. **5** shows a high level flowchart of a process of opening multiple network connections to the server in response to a request to open a single network connection;

FIG. **6** shows a flowchart of another process of opening multiple network connections to the server in response to a request to open a single network connection utilizing a cache;

FIG. **7**A and **7**B show flowcharts of a process of increasing performance of requests to get an object on the server if it has been modified utilizing a cache;

FIG. **8** shows a flowchart of a process of periodically refreshing information in the cache utilized in FIGS. **7**A and **7**B;

FIG. **9** is a block diagram of a browser connected to a Web server through network access equipment which utilizes a proxy server to increase performance;

FIG. **10** shows a flowchart of a process of directing an HTTP request to either the Web server or the proxy server depending on the request;

FIG. **11** is a block diagram of a browser connected to a Web server through network access equipment in which a client hook intercepts requests from the browser;

FIG. **12** shows a flowchart of a process of a client hook immediately responding that network connection has been opened in response to a request to open a network connection to the server;

FIG. **13** shows a flowchart of a process of an agent receiving a request from the client hook that includes a request from the client and an identifier for the server to which the request should be sent;

FIG. **14** shows a flowchart of a process of an agent storing a header and reconstructing another header from the differences between the headers; and

FIG. **15** shows a flowchart of a process of a client hook and agent increasing the performance of header transmission.

DESCRIPTION OF PREFERRED EMBODIMENTS

In the description that follows, the present invention will be described in reference to preferred embodiments that increase the performance of Web browsers utilizing a weak link to network access equipment. The present invention, however, is not limited to any particular embodiment or computer network. Therefore, the description the embodiments that follow is for purposes of illustration and not limitation.

FIG. **1** illustrates an example of a computer system used to execute software of an embodiment of the present invention. FIG. **1** shows a computer system **1** which includes a monitor **3**, screen **5**, cabinet **7**, keyboard **9**, and mouse **11**. Mouse **11** may have one or more buttons such as mouse buttons **13**. Cabinet **7** houses a CD-ROM drive **15**, a system memory and a hard drive (see FIG. **2**) which may be utilized to store and retrieve software programs incorporating computer code that implements the present invention, data for use with the present invention, and the like. Although a CD-ROM **17** is shown as an exemplary computer readable storage medium, other computer readable storage media including floppy disks, tape, flash memory, system memory, and hard drives may be utilized. Cabinet **7** also houses familiar computer components (not shown) such as a central processor, system memory, hard disk, and the like.

FIG. **2** shows a system block diagram of computer system **1** used to execute the software of an embodiment of the present invention. As in FIG. **1**, computer system **1** includes monitor **3** and keyboard **9**. Computer system **1** further includes subsystems such as a central processor **102**, system memory **104**, I/O controller **106**, display adapter **108**, removable disk **112** (e.g., CD-ROM drive), fixed disk **116** (e.g., hard drive), network interface **118**, and speaker **120**. Other computer systems suitable for use with the present invention may include additional or fewer subsystems. For example, another computer system could include more than one processor **102** (i.e., a multi-processor system) or a cache memory.

Arrows such as **122** represent the system bus architecture of computer system 1. However, these arrows are illustrative of any interconnection scheme serving to link the subsystems. For example, a local bus could be utilized to connect the central processor to the system memory and display adapter. Computer system **1** shown in FIG. **2** is but an example of a computer system suitable for use with the present invention. Other configurations of subsystems suitable for use with the present invention will be readily apparent to one of ordinary skill in the art.

Preferred embodiments of the invention increase the performance of Web browsers' (or clients') access to the Web on the Internet. FIG. **3** shows a diagram of multiple computers networked over the Internet. Computers **202, 204** and **206** are interconnected by the Internet **208**, which is a series of high-speed communications links between educational, research and commercial computer sites around the world. Internet computers use the TCP/IP as the communications protocol.

The Web utilizes the HTTP client/server protocol, which is a request-response protocol. HTTP transactions include four stages: connection, request, response, and disconnection. In the connection stage, the client attempts to open a network connection to the server. Unless otherwise specified, HTTP attempts to use port **80** on the server for this connection. Establishing a connection involves one round-trip time from the client to the server as the client requests to open a network connection and the server responds that a network connection has been opened. Although the discussion herein focuses on version 1.0 of HTTP, the invention is not limited to any version of HTTP or to HTTP specifically.

After a network connection is open, the client may send an HTTP request to the server in the request stage. A request stage involves one half of a round-trip time as the request goes from the client to the server. Once the server receives the request, the server responds by sending a response to the client in the response stage. As with the request, the response stage involves one half of a round-trip time as the response goes from the server to the client.

The disconnection stage closes the network connection to the server. This stage involves one half of a round-trip time and may occur many different ways. The server may close the connection after the response is sent or by the client by sending a Stop sequence (e.g., the user clicked on the Stop button in the browser or the Back/Forward buttons). Conventional browsers show each of the four stages on a status line on the screen.

The terms of "client" and "server" are relative terms. A client is an entity that is making a request to a server which typically responds back to the client. However, these labels are request-response specific and are not an indication that the entities' roles are fixed. In preferred embodiments, the client is a browser and the server is a Web server. The browser may be executed on a computer similar to the one shown in FIGS. **1** and **2**. The server may be similar but is typically a much more powerful system including faster subsystems and more storage capacity.

FIG. **4** is a block diagram of a browser connected to a Web server through network access equipment including an agent. The computer network shown includes a Web browser **252**, network access equipment **254** and a Web server **256**. The browser communicates over a link to the network access equipment via a network protocol (e.g., TCP/IP) stack **258**. The browser and network protocol stack reside on the client computer system. The network access equipment is typically an electronic box and may include some of the subsystems shown in FIG. **2**. The Web server resides on a server which is typically a remote computer system.

The network access equipment includes an agent **260**. The agent is a program that includes embodiments of the invention. The computer code for the agent may reside on any computer readable storage medium including dynamic random access memory, electrically erasable programmable read only memory, or flash memory just to name a few. In a preferred embodiment, the agent resides on a LanRover™ Access Switch remote access server available from Shiva Corporation, Bedford, Mass.

FIG. **5** shows a high level flowchart of a process of opening multiple network connections to the server in response to a request to open a single network connection. The process shown is executed by an agent on the network access equipment. At step **302**, the agent receives a client request to open a single network connection to the server.

In response to the client request to open a single network connection to the server, the agent sends multiple requests to the server to open multiple network connections to the server at step **304**. Thus, multiple network connections to the server are opened in response to a client request to open a single network connection. Preferably, the agent requests persistent network connections. Once one of the network connections is open, the agent will receive a response from the server and send that response to the client. The client will then issue a request to the server over the open network connection which will be received by the agent. At step **306**, the agent sends the following client request to the server using the open network connection.

Oftentimes, the agent will receive another client request to open a single network connection to the server. Since the agent previously opened multiple network connections, the agent responds immediately that a network connection is available, thus saving a round-trip time between the agent and server. The client then issues the following client request over the open network connection. The agent may store the open network connections in a cache, which will be described more detail in reference to FIG. **6**.

For simplicity, the discussion herein describes the interaction of the agent with a single client browser and a single Web server. However, in practice, the agent is typically in communication with multiple clients and multiple Web servers. The methods of the present invention are not isolated to increasing the performance of each individual client alone. For example, one client may open multiple network connections to the server by issuing a request to open a single network connection. Subsequently, another client may request to open a single network connection to the same server. The agent may then immediately grant a network connection to this client as a network connection has already been opened. Thus, the actions of one client may also result in an increase in performance of other clients. The agent preferably opens another network connection to the server to replace the one that has become used.

FIG. **6** shows a flowchart of another process of opening multiple network connections to the server in response to a request to open a single network connection utilizing a cache. In this embodiment, the agent maintains a cache of network connections to the server (or servers). At step **352**, the agent receives a client request to open a single network connection to the server.

The agent scans the cache for an open network connection to the server at step **354**. If an open network connection to the server is not available in the cache at step **356**, the agent sends a request to the server to open a network connection. Although this embodiment opens a single network connection at this point and makes a subsequent determination if connection caching improves performance, in another embodiment, the agent sends multiple requests to the server to open multiple network connections and bypasses the subsequent determination.

If an open network connection to the server is available in the cache, the agent sends a response to the client that a network connection is open. The client sends a client request to the server using the open network connection which the agent sends to the server at step **360**.

At step **362**, the agent determines if network connection caching for the server improves performance. This determination may be made from many factors including the number of times the agent has a "hit" in the cache, the overhead required to maintain the cache, whether the server is allowing the network connections to stay open in response to a request to "keep open" the connection (i.e., persistent connection), and the like. The agent may request that the server "keep open" the connection but honoring this request is at the discretion of the server.

The agent checks if all the network connections to the server in the cache are in use at step **364**. If connection caching does improve performance and all the network connections to the server are in use, the agent sends at least one request to the server to open a network connection at step **366**. Multiple network connections may be opened to the server if it has been determined that this improves performance. For example, it may be beneficial to have a predetermined number (e.g., user specified or determined by the agent as it monitors performance) of network connections open in the cache for a server. If the number of open connections is less than the predetermined number, the agent sends at least one request to open a network connection to the server.

In a preferred embodiment, the cache stores all the network connections and an indication of whether the network connection is open, in use or closed. As the agent opens network connections, they are marked as "open." When the agent receives a request to open a network connection to the server and there is an open network connection to the server in the cache, the agent marks the network connection as "in use" or "used." When the agent receives an indication from the server that a network connection in the cache has been closed, the agent marks the network connection as "closed."

In another embodiment, the cache stores only open network connections. Each time a network connection in the cache is either used or closed, the agent removes the network connection from cache. The agent may also issue a request to the server to open a network connection to replace a network connection removed from the cache.

In conventional network systems, when a client wants to get an object from the server, it takes two round-trip times between the client and server: one to open the connection and one to get the object. With the present invention, one round-trip time between the agent and server may be avoided, thus reducing the overall time to one and a half round-trip times between the client and server. This provides a significant increase in the performance of the client.

FIG. **7**A and **7**B show flowcharts of a process of increasing performance of requests to get an object on the server if it has been modified utilizing a cache. With HTTP, a client is able to request that the server send an object if it has not been modified since a specified time and date, which will be herein called a timestamp for convenience. More specifically, the header of an HTTP get message may include a header with fields specifying "If-Modified-Since" in one field and a timestamp in another field. If the server determines that the object has not been modified since the specified timestamp, the server does not need to send the object to the client but instead issues a not-modified **(304)** response.

Although this feature may be utilized by the browser to maintain its own cache, an agent of the present invention utilizes the feature to maintain a cache of information to further increase the performance of the computer network.

In general, it may take two client requests to set up the cache of information and a third client request to realize a performance increase. For completeness, the following assumes that browser does not have the desired object in its cache.

At step **402** in FIG. **7**A, the agent receives a client request to get an object from the server. The agent then sends the client request to the server. The agent receives the object from the server and sends it to the client at step **404**. Conventional browsers have a cache in which they store objects for future reference (e.g., when a user revisits the web page). The browser cache includes timestamps indicating the currency of the objects in the cache.

When the client desires an object in its cache, the browser sends a request to get the object if it has been modified since the timestamp specified in the browser cache. The agent receives this request and sends it to the server at step **406**.

At step **408**, the agent receives a response **(304)** from the server that the object has not been modified since the timestamp. The agent sends this response to the client. The agent stores an identifier for the object and the current timestamp in a cache (i.e., the timestamp of when the server indicated that the object had not changed) at step **410**. The current timestamp will be utilized as an estimate of the time at which the object remained unmodified. The cache may be a table including the address of the object (e.g., an identifier), the timestamp of the object in the browser's cache and the current timestamp. The agent does not need to store the object in the cache.

The agent receives a request to get the object from the server if it has been modified at step **412**. The agent determines if the request specifies an object in its caches by scanning the cache. As the previous client request described above requested this same object and received a not-modified response, the object is specified in the cache.

Now referring to FIG. **7**B, the agent determines if policy indicates the object in the browser's cache is sufficiently current at step **414**. The policy may be a comparison of the current timestamp to the timestamp in the cache of when the server last indicated that the object had not been modified. If the difference between these timestamps is within a predetermined amount of time, the object in the browser's cache is sufficiently current. The predetermined time may be set by a administrator or may be preset by the agent. Additional policy considerations may be applied. If the server does not change its contents often (e.g., as noticed by the agent), the amount of time may be lengthened. On the other hand, if the server does change its contents often (e.g., stock quotes), the amount of time may be shortened. Thus, the amount of time for an object still being current may be server, Web page or Uniform Resource Locator ("URL") specific.

At step **416**, the agent has determined that the object in the browser's cache is sufficiently current and the agent sends a not-modified response to the client. The agent responds to the client without sending a request to the server, thereby saving a round-trip time between the agent and server.

If the agent determines that the browser's cache is not sufficiently current, the agent sends a request to get the object form the server if it has been modified at step **418**. Thus, the agent sends the client request to the server. When the agent receives a response from the server, the agent sends the response to the client at step **420**. The agent updates the cache according to the response at step **422**. For example, the agent may store the current timestamp in the cache to indicate that at this point in time the server indicated that the object had not been modified. If a new copy of the object is received, the agent may also update the timestamp in the cache indicating the last time the browser received the object.

With the invention, the time for a client requesting an object if it has been modified may be reduced from one round-trip time between the client to the server to a round-trip time between the client and the agent (ignoring connection and disconnection times for the moment). Although the entries in the cache are client specific, the invention provides a significant performance increase for the clients when they issue a request to get an object if it has been modified.

FIG. **8** shows a flowchart of a process of periodically refreshing information in the cache utilized in FIGS. **7**A and **7**B. Periodically (e.g., using a timer), the agent gets an identifier for an object in the cache at step **452**. The agent then makes a determination of whether the object is sufficiently current at step **454**. This may be done by the agent making the same calculation as if a client requested to get the object if it has been modified.

If the object is not sufficiently current, the agent sends a request to the server to get the object if it has been modified at step **456**. This request originates from the agent and not the client. The agent then updates its cache at step **458** depending on the response from the server. If the server responds that the object has not been modified, the agent may update the estimate of the time at which the object remained unmodified. Otherwise, if the server sends a new copy of the object, the agent typically discards the new object and updates the cache to indicate the object has been modified. In other embodiments, the agent may store the new copy of the object in order to fulfill future client requests.

At step **460**, the agent determines if there is another object identified in the cache. If there is, the agent tries to update the cache for that object. By periodically updating the cache, the performance of the client browser will increase as more round-trip times between the agent and server may be eliminated.

FIG. **9** is a block diagram of a browser connected to a Web server through network access equipment which utilizes a proxy server to increase performance. The computer network shown includes a Web browser **502**, network access equipment **504** and a Web server **506**. The browser communicates over a link to the network access equipment via a network protocol stack **508**. The browser and network protocol stack reside on the client computer system. The Web server resides on a server which is typically a remote computer system.

The network access equipment includes an agent **510**. The agent shown is a program that receives HTTP requests and directs them to either the Web server or a proxy server **512**. The agent will typically receives messages in a number of protocols but the discussion herein will focus on HTTP messages. The proxy server is a computer system that stores information available from the Web server. In general, it may be quicker to access information from the proxy server instead of the Web server.

Although the use of proxy servers is known, conventional systems require the client to specify whether an HTTP message be sent to the Web server or the proxy server. With the present invention the client need not explicitly specify the proxy server to gain an increase in performance resulting from use of the proxy server. The agent of the present invention sends the HTTP requests to either the Web server or proxy server based on an analysis of the HTTP request.

FIG. **10** shows a flowchart of a process of directing an HTTP request to either the Web server or the proxy server depending on the request. At step **552**, the agent receives an HTTP request from the client. The agent analyzes the HTTP request at step **554**. The analysis may include a determination of whether the request gets information or posts information. Requests that post information may be sent to the Web server. However, requests that get information may be sent to the proxy server. There may also be other factors including determining if this information is likely to reside on the proxy server.

If it is determined that the HTTP request may be serviced by the proxy server at step **556**, the agent sends the HTTP request to the proxy server at step **558**. The agent may also need to translate the request to a different protocol before it is sent to the proxy server. Otherwise, the agent sends the HTTP request to the Web server at step **560**.

The invention allows a client to obtain the benefits of a proxy server without needing to be modified to send requests explicitly to the proxy server. Thus, the proxy server may be changed or otherwise modified by effecting changes to the network access equipment while the client remains unchanged.

FIG. **11** is a block diagram of a browser connected to a Web server through network access equipment in which a client hook intercepts requests from the browser. The computer network shown includes a Web browser **602**, network access equipment **604** and a Web server **606**. The browser communicates over a link to the network access equipment via a network protocol stack **608**. On the client computer system with the browser and network protocol stack is a client hook **610**, the client hook intercepts calls between the browser and the network protocol stack.

In preferred embodiment, the client hook intercepts calls between the browser and the network protocol stack utilizing DLL chaining. For example, the dynamic link library WINSOCK.DLL is renamed to WINSOCKZ.DLL. A new WINSOCK.DLL is installed on the client computer system that has routines with the same name as in the original WINSOCK.DLL. However, the new WINSOCK.DLL has instructions in the routines (i.e., the client hook) to intercept calls before they are executed. In many instances, the routines in WINSOCK.DLL call the routines in WINSOCKZ.DLL at some point in the routine.

The network access equipment includes an agent **612**. The agent shown is a program that receives HTTP requests from the client hook. The client hook and agent communicate in such a way to increase performance of the computer network without requiring a modification of the client. Accordingly, a user is free to select the browser of his or her choice and still receive a significant performance increase.

In general, the client hook intercepts HTTP requests from the client to the server. The client hook modifies the HTTP requests from the client and sends the modified requests to the agent. The agent receives the modified requests and reconstructs the original HTTP requests from the client according to the modified requests. The agent then sends the HTTP requests from the client to the server. There is no requirement that the client hook and agent communicate via HTTP. Nevertheless, it is the communication between the client hook and the agent increases performance of communication between the client and the server.

The computer network shown in FIG. **11** may be utilized to increase performance of many procedures. For example, the procedure of opening a network connection to the server may be improved. Additionally, the procedure of sending headers within requests to the server may enhanced. These are but a couple examples of the present invention which will be described in more detail in reference to FIGS. **12–15**.

FIG. 12 shows a flowchart of a process of a client hook immediately responding that network connection has been opened in response to a request to open a network connection to the server. At step 622, the client hook intercepts a client request to open a network connection to the server. The client hook immediately responds to the client that a dummy network connection has been opened at step 624. The dummy network connection is not an actual network connection but allows the client to proceed with the next client request.

The client hook intercepts a client request to the server that specifies the dummy network connection at step 626. At step 628, the client hook sends the client request and an identifier for the server to the agent. The identifier for the server (e.g., the address) is obtained from the client request to open a network connection. As the client hook and agent are in communication within the computer network, there is no requirement that the messages between the two conform to HTTP. Thus, the actual protocol utilized may be optimized for the actual link.

FIG. 13 shows a flowchart of a process of an agent receiving a request from the client hook that includes a request from the client and an identifier for the server to which the request should be sent. At step 652, the agent receives a request from the client hook which includes an identifier for the server. The agent receives the request from the client hook without necessarily first receiving an HTTP client request to open a network connection to the server.

The agent generates and sends an HTTP request to the server to open a network connection at step 654. Preferably, the agent requests a persistent network connection. The server is identified by the identifier received from the client hook. Once the agent receives a response from the server that a network connection is open, the agent generates and sends the client request in the form of an HTTP request to the server at step 656.

The agent may send an HTTP request to open a network connection to the server. The agent may also maintain a cache of network connections as was described in reference to FIG. 6. In this manner, a round-trip time between the agent and server may be eliminated.

The invention increases performance in many ways. A round-trip time between the client and agent may be eliminated when opening a network connection to the server. This may be especially significant because this link may be the weak link in the computer network. Additionally, the protocol between the client hook and agent is not restricted to HTTP so it may more optimized.

FIG. 14 shows a flowchart of a process of an agent storing a header and reconstructing another header from the differences between the headers. Initially, the client hook intercepts client requests and sends them to the agent. As will be described below, the communication between the client hook and agent is preferably not HTTP as it is optimized. Additionally, although the embodiment described is directed to headers in the requests, the invention is applicable to any information within the requests.

When the agent receives a request from the client hook, the agent stores the header at step 672. The agent then generates and sends a corresponding HTTP request to the server. At step 674, the agent receives another request from the client hook that includes differences between the previous header and this header. The differences between headers is not currently a standard format of headers in HTTP.

The agent reconstructs the header for the current request from the client hook utilizing the stored header and the differences at step 676. At step 678, the agent uses the reconstructed header in generating and sending a corresponding HTTP request to the server.

The header typically includes information about the browser (e.g., name and version number), acceptable data formats, and the like similar to a Multipurpose Internet Mail Extensions ("MIME") header. Accordingly, much of the header does not change from request to request. With the invention, an HTTP request may be reduced from several hundred bytes to a request that is less than twenty bytes. This is especially significant as the link between the client hook and agent is typically the weak link in the computer network. The following describes an embodiment of this process in more detail.

FIG. 15 shows a flowchart of a process of a client hook and agent increasing the performance of header transmission. At step 702, the client hook intercepts a client request to the server that includes a header, which the client hook stores. The client request is an HTTP request and the client hook utilize a more optimized protocol (i.e., non-HTTP) when it sends the client request to the agent.

The client hook sends the client request to the agent for transmission to the server at step 704. Once the agent receives the client request, the agent stores a copy of the header in the client request at step 706. If the client request is non-HTTP, the agent generates a corresponding HTTP client request. The agent sends the client request to the server at step 708.

At step 710, the client hook intercepts a client request to the server that includes a header. The client hook modifies the client request to include a header that specifies differences between this header and the previous header at step 712. Thus, the client request will contain the differences or deltas between the headers.

The agent receives the modified client request at step 714. With the modified client request, the agent reconstructs the header from the stored header and the differences between the headers at step 716. The agent generates an HTTP request that corresponds to the client request and includes the reconstructed header. The agent sends the client request to the server at step 718.

The invention increases performance in many ways. A round-trip time between the client and agent may be eliminated when opening a network connection to the server. This may be especially significant because the link between the client hook and agent may be substantially slower than the link between the agent and server. Additionally, the protocol between the client hook and agent is not restricted to HTTP so it may be more optimized.

While the above is a complete description of preferred embodiments of the invention, various alternatives, modifications and equivalents may be used. It should be evident that the present invention is equally applicable by making appropriate modifications to the embodiments described above. For example, although the embodiments have been described individually, many of the embodiments may be combined to further increase performance. Therefore, the above description should not be taken as limiting the scope of the invention which is defined by the metes and bounds of the appended claims along with their full scope of equivalents.

What is claimed is:

1. In a computer network, a method executed by an agent in the computer network between clients and a server for increasing performance between the clients and the server, the method comprising the steps of:

receiving a first request from a client to open a single network connection to the server;

sending a plurality of requests to the server to open a plurality of network connections to the server;

receiving a second request from the client;

sending the second request to the server using one of the plurality of network connections;

wherein the plurality of network connections to the server are opened in response to the first request from the client to open a single network connection.

2. The method of claim **1**, further comprising the steps of:

receiving a third request from a client to open a single network connection to the server;

sending a response to the client that a network connection is open;

receiving a fourth request from the client; and

sending the fourth request to the server using one of the plurality of network connections previously obtained in response to the first request.

3. The method of claim **2**, further comprising the step of sending a request to the server to open a network connection to the server in order to replace the one of the plurality of network connections being used.

4. The method of claim **1**, further comprising the step of storing the plurality of network connections in a cache of network connections.

5. The method of claim **4**, further comprising the steps of:

receiving a third request from a client to open a single network connection to the server;

scanning the cache to determine if there is an open network connection to the server in the cache; and

if there is an open network connection in the cache, sending a response to the client that a network connection is open, whereby the open network connection becomes used.

6. The method of claim **5**, further comprising the steps of:

determining if network connection caching increases performance between the clients and the server; and

if network caching increases performance, sending a request to the server to open a network connection to the server in order to store another open network connection in the cache to replace the used network connection.

7. The method of claim **4**, further comprising the steps of:

determining the number of open network connections to the server stored in the cache; and

if the number of open network connections is less than a predetermined number, sending a request to the server to open a network connection to the server in order to store another open network connection in the cache.

8. The method of claim **4**, further comprising the steps of:

determining if an open network connection to the server in the cache has been closed; and

if there is a closed network connection in the cache, sending a request to the server to open a network connection to the server in order to store an open network connection in the cache.

9. The method of claim **5**, further comprising the step of removing the used network connection from the cache.

10. The method of claim **5**, further comprising the step of sending a request to the server to open a network connection to the server in order to store another open network connection in the cache to replace the used network connection.

11. The method of claim **1**, wherein the client is a World Wide Web browser.

12. A computer network, comprising:

a client computer running a Web browser;

a Web server networked to the client computer;

a proxy server computer networked to the client computer for storing information available on the Web server; and

network access equipment, networked between the client computer and the Web and proxy servers, including an agent that receives an HTTP request from the Web browser to open a single network connection to the server and sends a plurality of requests to the server to open a plurality of network connections to the server;

wherein the plurality of network connections to the server are opened in response to the HTTP request from the Web browser to open a single network connection.

13. In a computer network, a method executed by an agent in the computer network between clients and a server for increasing performance between the clients and the server, the method comprising the steps of:

receiving a first request from a client to get an object from the server if the object has been modified after a specific timestamp;

sending the first request to the server;

receiving a first response from the server that the object has not been modified after the specific timestamp;

sending the first response to the client;

storing an identifier for the object and a timestamp in a cache;

receiving a second request from the client to get the object from the server if the object has been modified after the specific timestamp; and

if the timestamp stored in the cache is within a predetermined amount of time from the current time, sending a second response to the client that the object has not been modified after the specific timestamp without sending the second request to the server.

14. The method of claim **13**, wherein the storing step includes the steps of storing a location of the object as the identifier, storing the specific timestamp, and storing the timestamp as the current time in order to estimate at what time the object remained unmodified.

15. The method of claim **14**, further comprising the step of periodically sending requests to the server to get objects identified in the cache if the object has been modified after the specific timestamp in order to update the timestamp in the cache.

16. The method of claim **13**, further comprising the step of setting the predetermined amount of time.

17. The method of claim **13**, wherein the client is a World Wide Web browser.

18. A computer network, comprising:

a client computer running a Web browser;

a Web server networked to the client computer;

a proxy server computer networked to the client computer for storing information available on the Web server; and

network access equipment, networked between the client computer and the Web and proxy servers, including an agent that stores identifiers and timestamps for objects so that when the agent receives a request from the Web browser to get an object from the server if the object has been modified after a specific timestamp, the agent responds to the request without sending a request to the Web server.

19. In a computer network, a method executed by an agent in the computer network between a client and a Web and proxy servers for increasing performance between the client and the Web server, comprising the steps of:

receiving an HTTP request from a client; and

sending the HTTP request to either the Web server or the proxy server depending on the HTTP request, the proxy server storing information available on the Web server;

wherein the client does not need to be modified to utilize the proxy server.

20. The method of claim 19, wherein if the HTTP request may be serviced by the proxy server, the HTTP request is sent to the proxy server, and otherwise the HTTP request is sent to the Web server.

21. The method of claim 19, wherein if the HTTP request is to post information to the server, the HTTP request is sent to the Web server, and otherwise the HTTP request is sent to the proxy server.

22. The method of claim 19, further comprising the step of translating the HTTP request to a different protocol before the HTTP request is sent to the proxy server.

23. The method of claim 19, wherein the client is a World Wide Web browser.

24. A computer network, comprising:

a client computer running a Web browser;

a Web server networked to the client computer;

a proxy server computer networked to the client computer for storing information available on the Web server; and

network access equipment, networked between the client computer and the Web and proxy servers, including an agent that receives HTTP requests and sends the HTTP requests to either the Web server or the proxy server depending on each HTTP request;

wherein software on the client computer does not need to be modified to utilize the proxy server.

25. The computer network of claim 24, wherein HTTP requests that may be serviced by the proxy server are sent to the proxy server, otherwise the HTTP requests are sent to the Web server.

26. The computer network of claim 24, wherein HTTP requests that post information to the server are sent to the Web server, otherwise the HTTP requests are sent to the proxy server.

27. The computer network of claim 24, wherein the HTTP requests are translated to a different protocol before the HTTP requests are sent to the proxy server.

28. In a computer network, a method for increasing performance between a client on a client computer and a server utilizing a client hook on the client computer and an agent between the client computer and the server, comprising the steps of:

the client hook intercepting requests from the client to the server;

the client hook modifying the requests from the client;

the client hook sending the modified requests to the agent;

the agent reconstructing the requests from the client according to the modified requests; and

the agent sending the requests from the client to the server;

wherein communication between the client hook and the agent increases performance of communication between the client and the server.

29. The method of claim 28, further comprising the steps of:

the client hook intercepting a first request from the client to open a network connection to the server, the first request including an identifier for the server;

the client hook immediately responding that a network connection to the server has been opened to the server and storing the identifier of the server;

the client hook intercepting a second request from the client to be sent over the opened network connection to the server; and

the client hook sending the second request and the identifier of the server to the agent.

30. The method of claim 29, further comprising the steps of:

the agent sending a third request to open a network connection to the server identified by the identifier; and

the agent sending the second request to the server over an open network connection.

31. The method of claim 29, further comprising the steps of:

the agent identifying an open network connection to the server in a cache; and

the agent sending the second request to the server over the open network connection.

32. The method of claim 29, further comprising the steps of:

without first receiving a request from the client to open a network connection to the server, the agent receiving a first request from the client to the server and an identifier for the server;

utilizing the identifier for the server, the agent sending a second request to the server to open a network connection; and

the agent sending the first request to the server over an open network connection.

33. The method of claim 28, further comprising the steps of:

the agent storing first information included in a first request from the client to the server;

the agent receiving a second request from the client to the server that includes differences between the first information and second information of the second request instead of the second information;

the agent reconstructing the second information from the stored first information and the differences between the first and second information; and

the agent sending the second request to the server including the reconstructed second information.

34. The met hod of claim 28, further comprising the steps of:

the client hook intercepting a first request from the client to the server that includes first information;

the client hook sending the first request to the agent for sending to the server;

the agent storing a copy of the first information;

the agent sending the first request to the server;

the client hook intercepting a second request from the client to the server that includes second information;

the client hook modifying the second request to include differences between the first and second information instead of the second information;

the agent receiving the modified second request from the client;

the agent reconstructing the second information from the stored first information and the differences between the first and second information; and the agent sending the second request to the server including the reconstructed second information.

35. The method of claim **28**, wherein the network link between the client computer and the agent is substantially slower than the network link between the agent and the server.

36. In a computer network, a method for increasing performance between a client on a client computer and a server utilizing a client hook on the client computer and an agent between the client computer and the server, comprising the steps of:

the client hook intercepting a first request from the client to open a network connection to the server, the first request including an identifier for the server;

the client hook immediately responding that a network connection to the server has been opened to the server and storing the identifier of the server;

the client hook intercepting a second request from the client to be sent over the opened network connection to the server; and

the client hook sending the second request and the identifier of the server to the agent.

37. The method of claim **36**, further comprising the steps of:

the agent sending a third request to open a network connection to the server identified by the identifier; and

the agent sending the second request to the server over an open network connection.

38. The method of claim **36**, further comprising the steps of:

the agent identifying an open network connection to the server in a cache; and

the agent sending the second request to the server over the open network connection.

39. The method of claim **36**, wherein the second request to the server to open a network connection includes a request to keep the network connection open.

40. In a computer network, a method executed by an agent in the computer network between a client and a server for increasing performance between the client and the server, the method comprising the steps of:

without first receiving a request from the client to open a network connection to the server, receiving a first request from the client to the server and an identifier for the server;

utilizing the identifier for the server, sending a second request to the server to open a network connection; and

sending the first request to the server over an open network connection.

41. The method of claim **40**, wherein the second request to the server to open a network connection includes a request to keep the network connection open.

42. The method of claim **40**, wherein the client is a World Wide Web browser.

43. In a computer network, a method executed by an agent in the computer network between a client and a server for increasing performance between the client and the server, the method comprising the steps of:

storing first information included in a first request from the client to the server;

receiving a second request from the client to the server that includes differences between the first information and second information of the second request instead of the second information;

reconstructing the second information from the stored first information and the differences between the first and second information; and

sending the second request to the server including the reconstructed second information.

44. The method of claim **43**, wherein the first and second requests are HTTP requests.

45. In a computer network, a method for increasing performance between a client on a client computer and a server utilizing a client hook on the client computer and an agent between the client computer and the server, comprising the steps of:

the client hook intercepting a first request from the client to the server that includes first information;

the client hook sending the first request to the agent for sending to the server;

the agent storing a copy of the first information;

the agent sending the first request to the server;

the client hook intercepting a second request from the client to the server that includes second information;

the client hook modifying the second request to include differences between the first and second information instead of the second information;

the agent receiving the modified second request from the client;

the agent reconstructing the second information from the stored first information and the differences between the first and second information; and

the agent sending the second request to the server including the reconstructed second information.

46. The method of claim **45**, wherein the client hook intercepts requests from the client utilizing dynamic link library chaining.

47. The method of claim **45**, wherein the first and second requests are HTTP requests.

48. The method of claim **45**, wherein the client is a World Wide Web browser.

* * * * *

(C) Evidence for Claims 5-9, 11, 13, 24 and 25 – Relied Upon

The following item (1) listed below is hereby entered as evidence relied upon by the Examiner as to grounds of rejection for claims 5-9, 11, 13, 24 and 25 to be reviewed on appeal. Also listed for each item is where said evidence was entered into the record by the Examiner.

(1) Copy of US Patent Number 6,609,154 B1 ("Fuh et al"). This evidence was entered into the record by the Examiner on page 6 at 9, page 8 at 10, and page 10 at 12, of the Office Action mailed 08/10/2006.

Copies of all References follows.

//

US006609154B1

(12) **United States Patent**
Fuh et al.

(10) **Patent No.: US 6,609,154 B1**
(45) **Date of Patent: *Aug. 19, 2003**

(54) LOCAL AUTHENTICATION OF A CLIENT AT A NETWORK DEVICE

(75) Inventors: **Tzong-Fen Fuh**, Fremont, CA (US); **Serene H. Fan**, Palo Alto, CA (US); **Diheng Qu**, Santa Clara, CA (US)

(73) Assignee: **Cisco Technology, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **10/264,655**

(22) Filed: **Oct. 3, 2002**

Related U.S. Application Data

(63) Continuation of application No. 09/347,433, filed on Jul. 2, 1999, now Pat. No. 6,463,474.

(51) **Int. Cl.**^{7} **G06F 15/173**

(52) **U.S. Cl.** **709/225**

(58) **Field of Search** 709/225, 229, 709/222, 223; 713/200, 201

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,991,807 A	*	11/1999	Schmidt et al.	709/225
6,182,142 B1	*	1/2001	Win et al.	709/229
6,219,706 B1	*	4/2001	Fan et al.	709/225
6,233,576 B1	*	5/2001	Lewis	707/9
6,233,618 B1	*	5/2001	Shannon	709/229
6,292,798 B1	*	9/2001	Dockter et al.	707/9
6,292,904 B1	*	9/2001	Broomhall et al.	714/1

* cited by examiner

Primary Examiner—Glenton B. Burgess
Assistant Examiner—Kimberly Flynn
(74) *Attorney, Agent, or Firm*—Hickman Palermo Truong & Becker LLP

(57) ABSTRACT

A method and apparatus that provide network access control are disclosed. In one embodiment, a network device is configured to intercept network traffic initiated from a client and directed toward a network resource, and to locally authenticate the client. Authentication is carried out by comparing information identifying the client to authentication information stored in the network device. In one embodiment, an authentication cache in the network device stores the authentication information. If the client identifying information is authenticated successfully against the stored authentication information, the network device is dynamically re-configured to allow network traffic initiated by the client to reach the network resource. If local authentication fails, new stored authentication is created for the client, and the network device attempts to authenticate the client using a remote authentication server. If remote authentication is successful, the local authentication information is updated so that subsequent requests can authenticate locally. As a result, a client may be authenticated locally at a router or similar device, reducing network traffic to the authentication server.

29 Claims, 9 Drawing Sheets

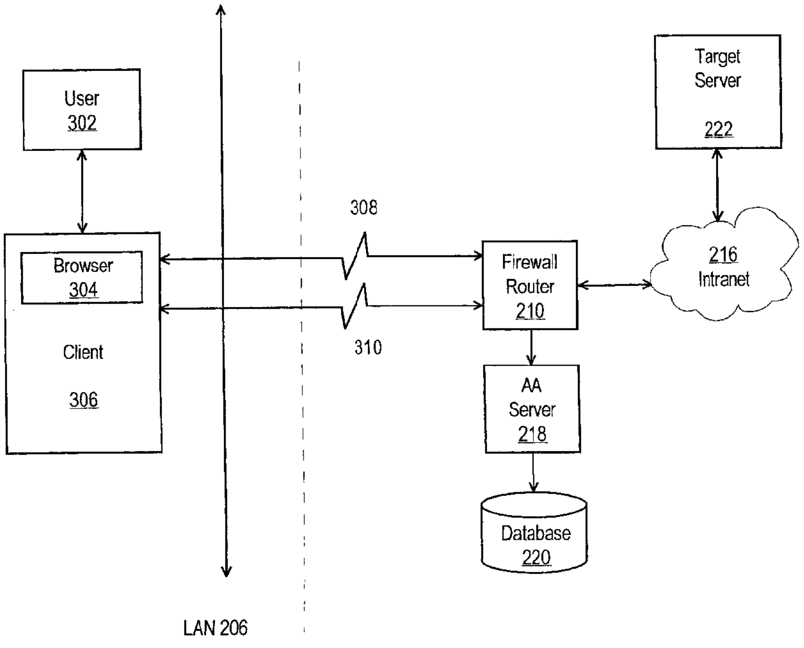
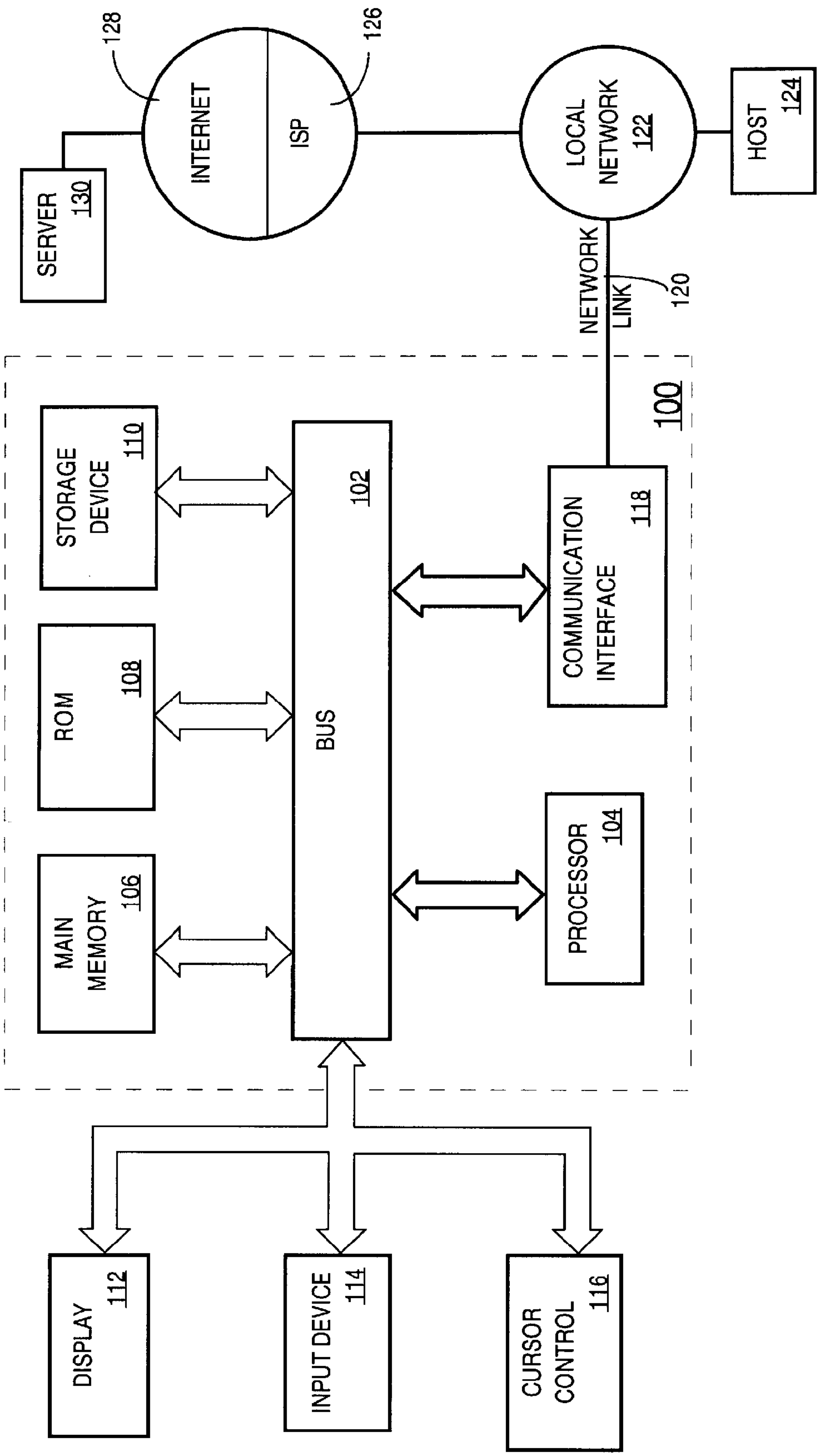


FIG. 1


SERVER
130
INTERNET
128
ISP
126
LOCAL
NETWORK
122
HOST
124
NETWORK
LINK
120
100
STORAGE
DEVICE
110
ROM
108
MAIN
MEMORY
106
BUS
102
COMMUNICATION
INTERFACE
118
PROCESSOR
104
DISPLAY
112
INPUT DEVICE
114
CURSOR
CONTROL
116

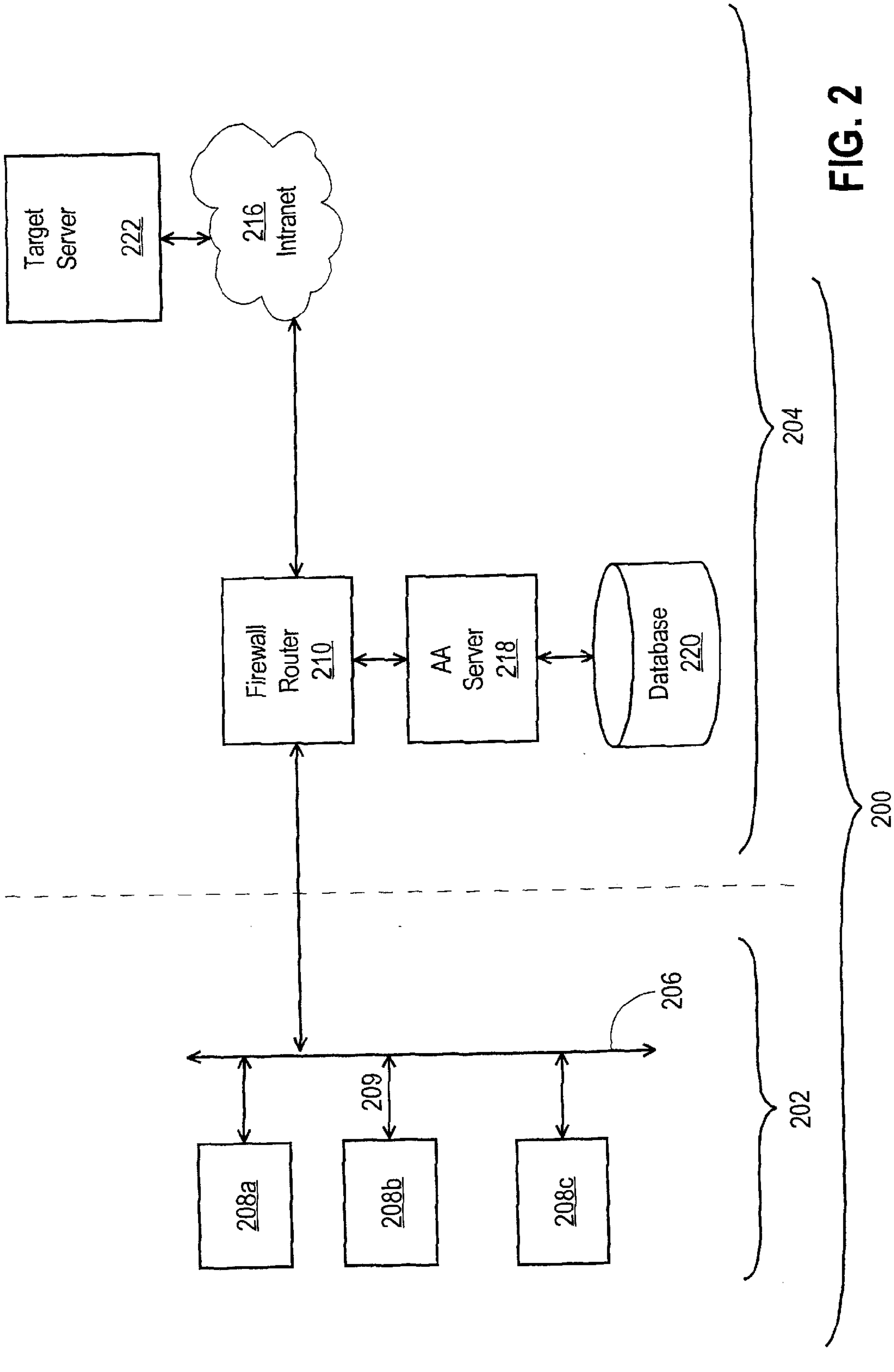

Target Server 222
216 Intranet
Firewall Router 210
AA Server 218
Database 220
204
200
206
209
208a
208b
208c
202

FIG. 2

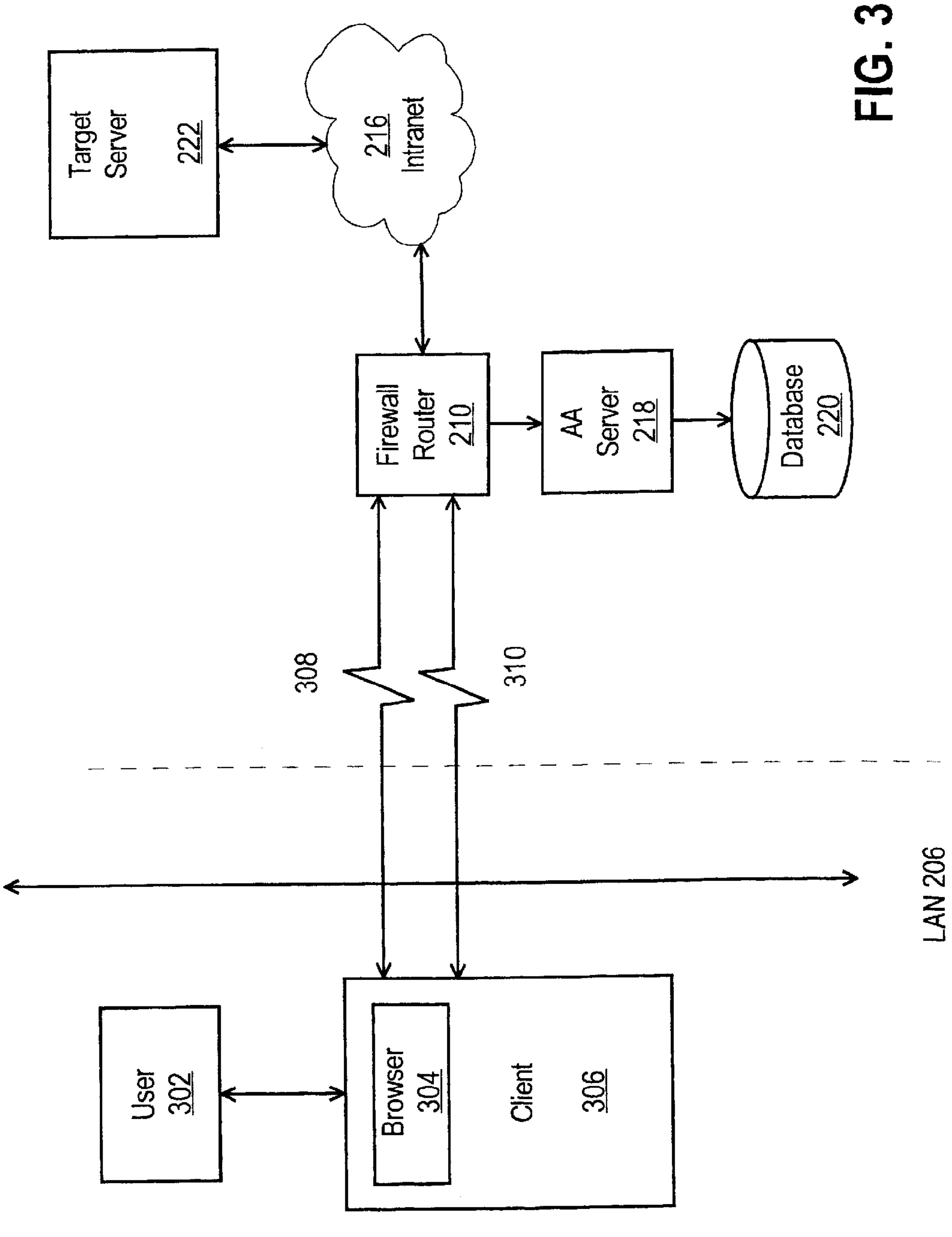

Target Server 222
216 Intranet
Firewall Router 210
AA Server 218
Database 220
308
310
LAN 206
User 302
Browser 304
Client
306

FIG. 3

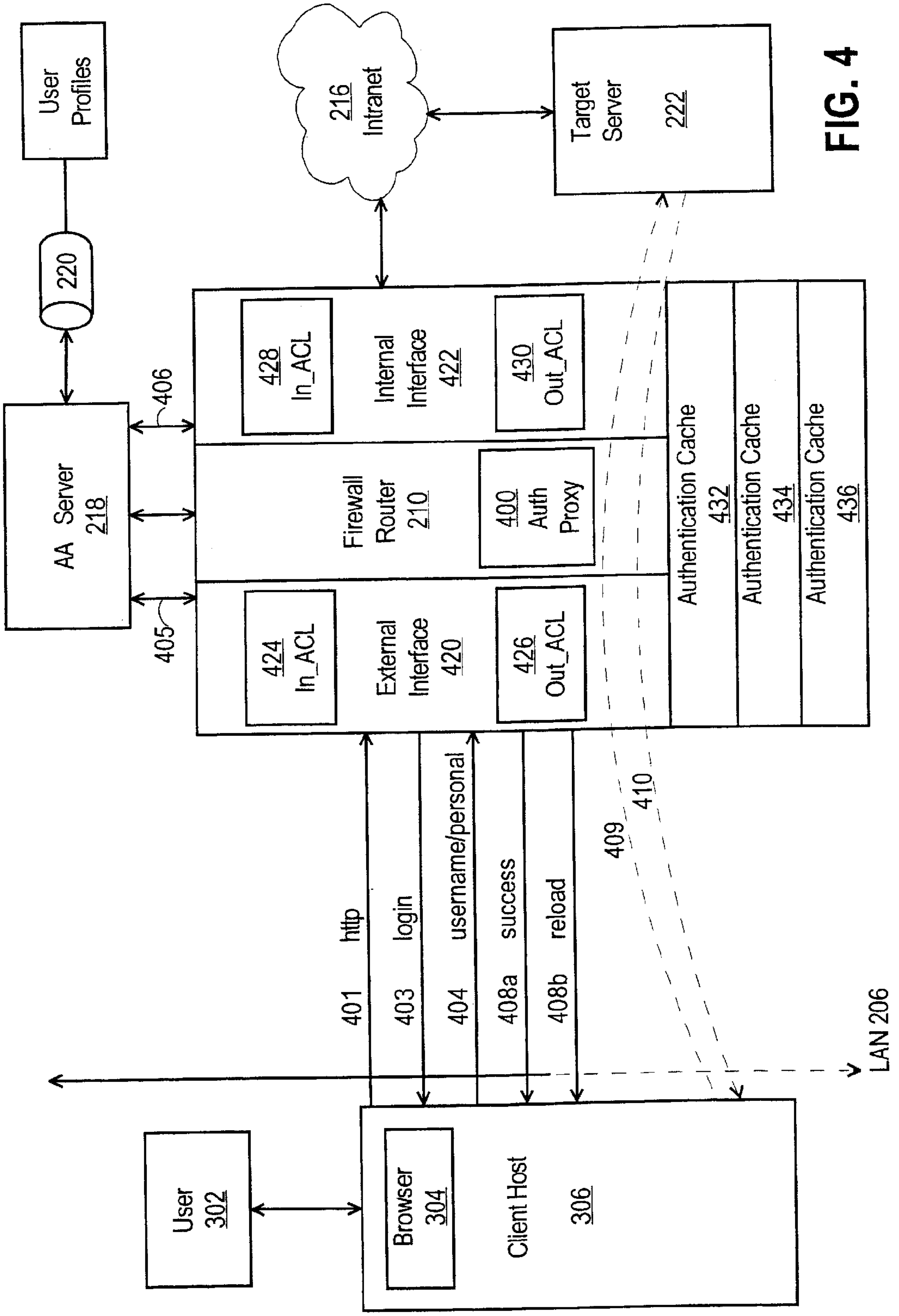
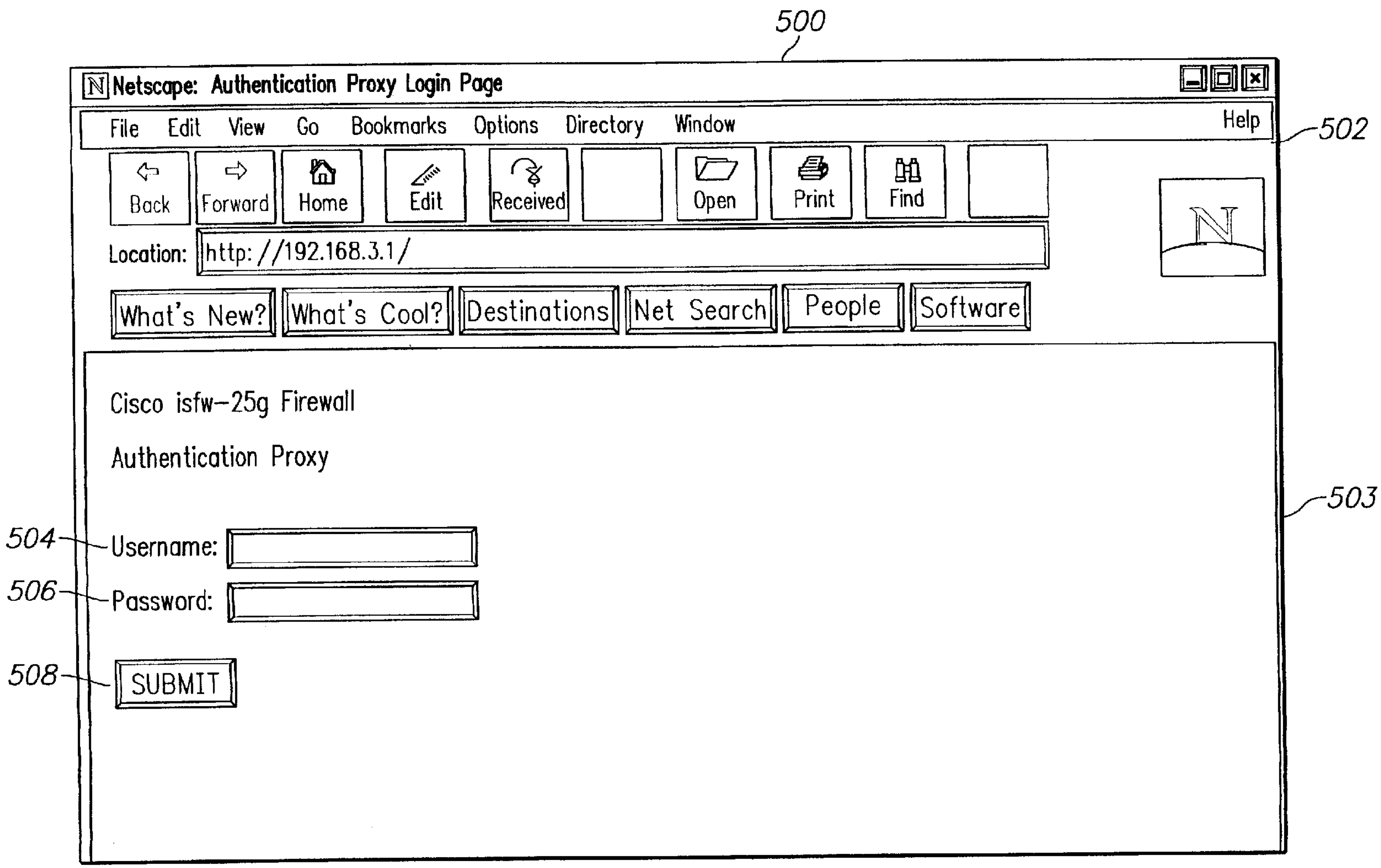

User Profiles
220
AA Server 218
406
405
216 Intranet
Target Server 222
428 In_ACL
Internal Interface 422
430 Out_ACL
Firewall Router 210
400 Auth Proxy
424 In_ACL
External Interface 420
426 Out_ACL
Authentication Cache 432
Authentication Cache 434
Authentication Cache 436
401 http
403 login
404 username/personal
408a success
408b reload
409
410
LAN 206
User 302
Browser 304
Client Host 306

FIG. 4



Netscape: Authentication Proxy Login Page

Location: http://192.168.3.1/

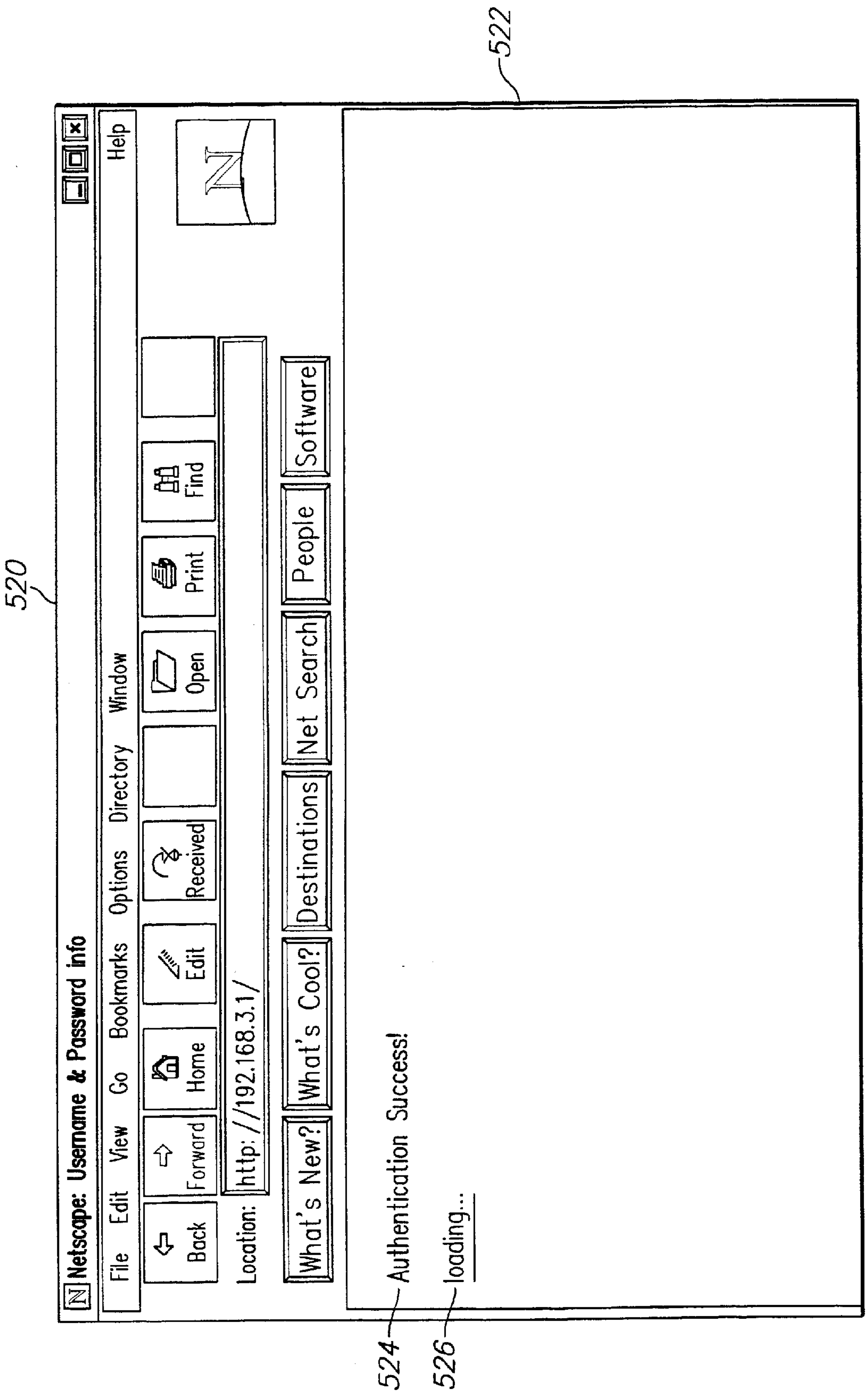
Cisco isfw-25g Firewall

Authentication Proxy

Username:

Password:

FIG. 5A


Authentication Success!

loading...

FIG. 5B

FIG. 6

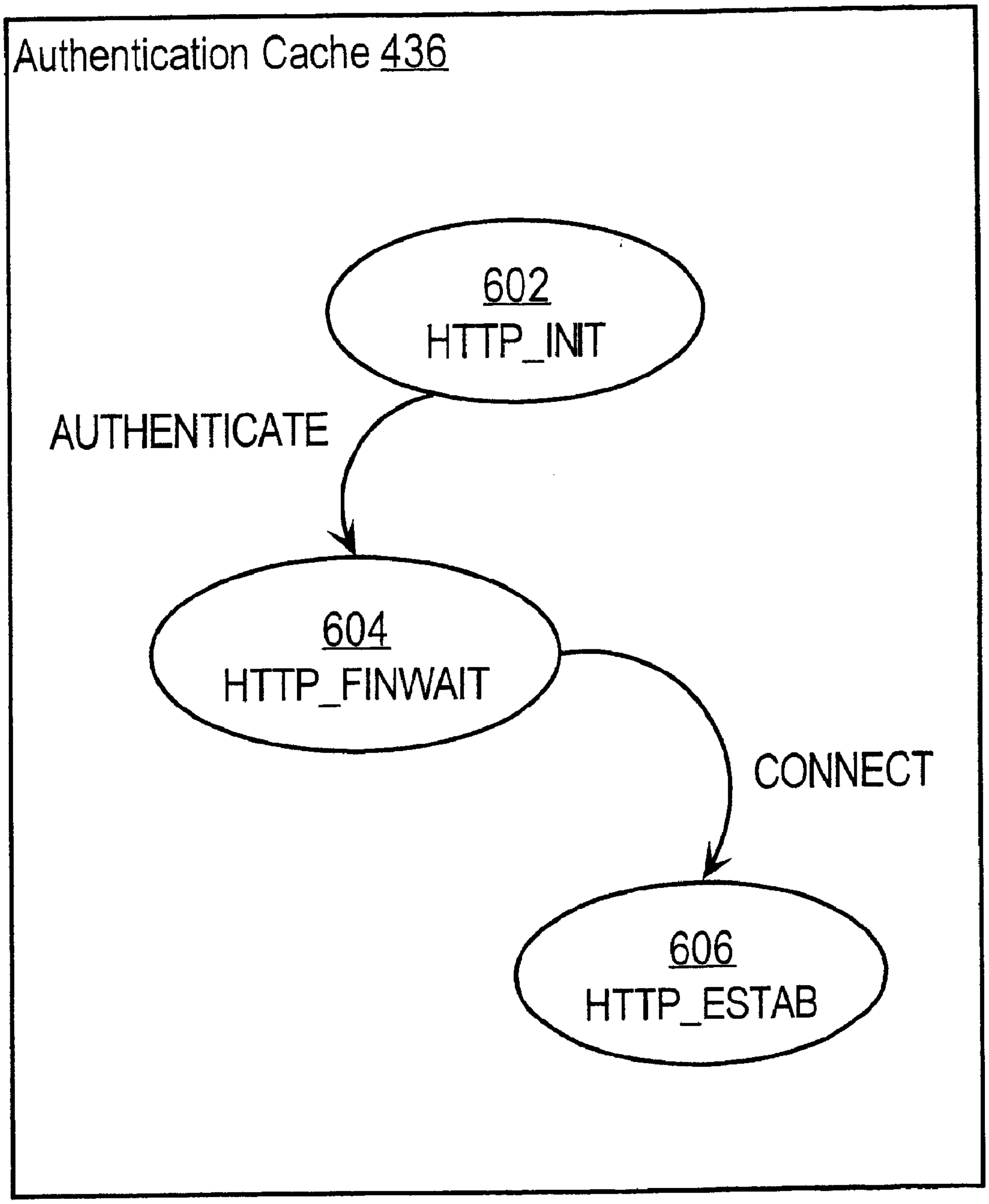

Authentication Cache 436
602
HTTP_INIT
AUTHENTICATE
604
HTTP_FINWAIT
CONNECT
606
HTTP_ESTAB

FIG. 7A

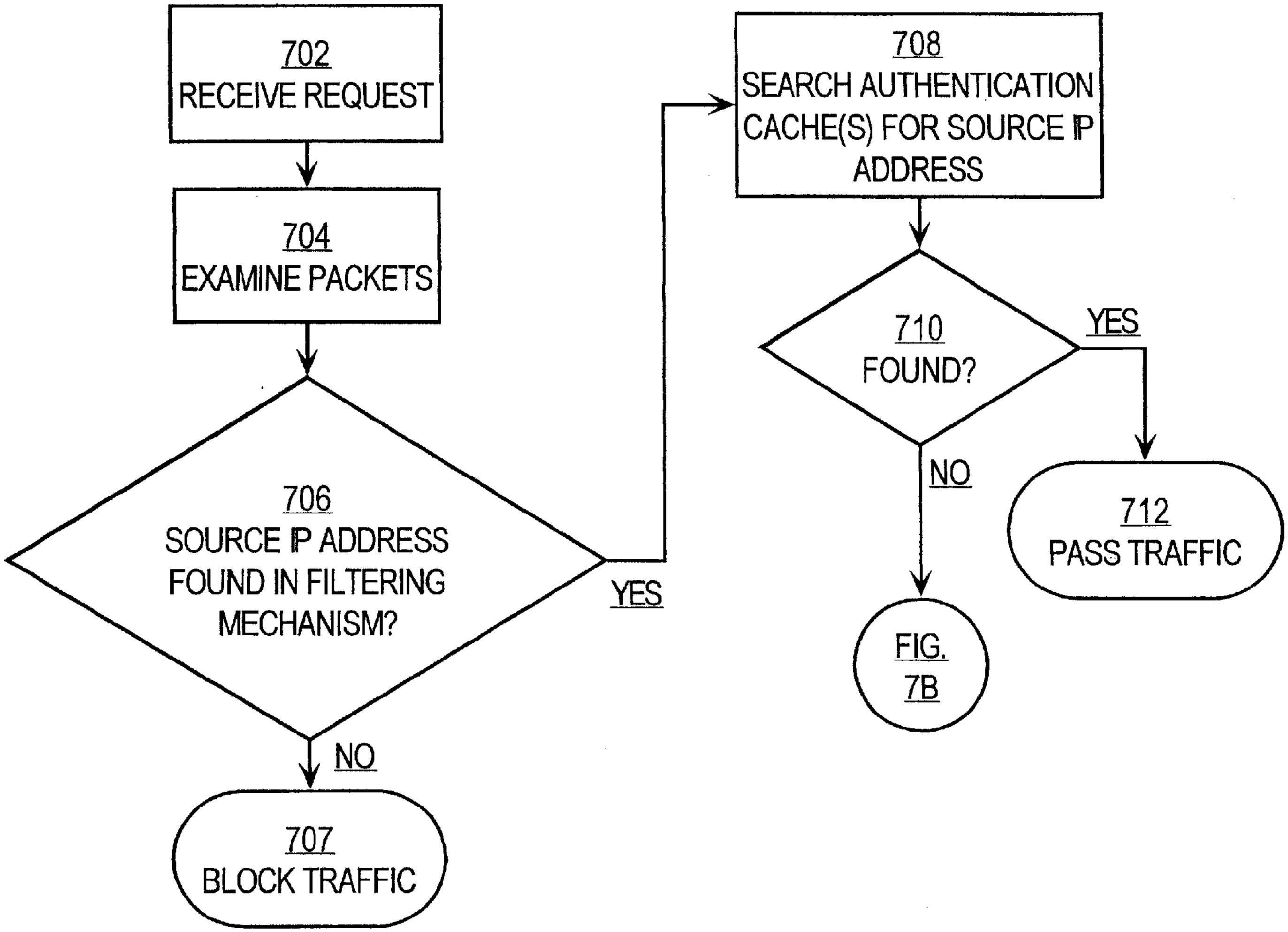
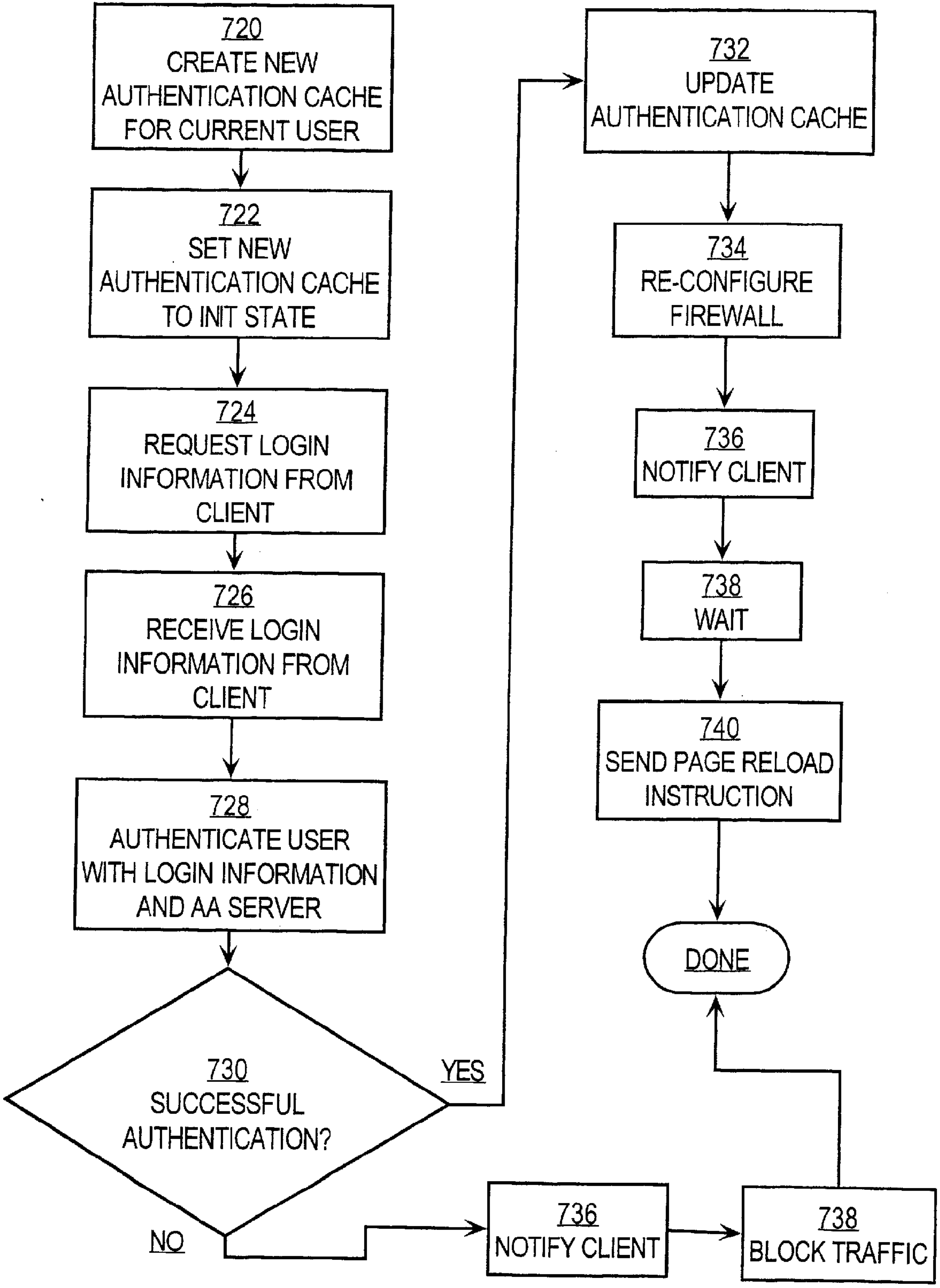

702
RECEIVE REQUEST
704
EXAMINE PACKETS
706
SOURCE IP ADDRESS FOUND IN FILTERING MECHANISM?
YES
NO
707
BLOCK TRAFFIC
708
SEARCH AUTHENTICATION CACHE(S) FOR SOURCE IP ADDRESS
710
FOUND?
YES
NO
712
PASS TRAFFIC
FIG.
7B

FIG. 7B


720
CREATE NEW AUTHENTICATION CACHE FOR CURRENT USER
722
SET NEW AUTHENTICATION CACHE TO INIT STATE
724
REQUEST LOGIN INFORMATION FROM CLIENT
726
RECEIVE LOGIN INFORMATION FROM CLIENT
728
AUTHENTICATE USER WITH LOGIN INFORMATION AND AA SERVER
730
SUCCESSFUL AUTHENTICATION?
YES
NO
732
UPDATE AUTHENTICATION CACHE
734
RE-CONFIGURE FIREWALL
736
NOTIFY CLIENT
738
WAIT
740
SEND PAGE RELOAD INSTRUCTION
DONE
736
NOTIFY CLIENT
738
BLOCK TRAFFIC

LOCAL AUTHENTICATION OF A CLIENT AT A NETWORK DEVICE

CROSS-REFERENCE TO RELATED APPLICATIONS; PRIORITY CLAIM

This application claims priority under 35 U.S.C. §120 as a Continuation of prior application Ser. No. 09/347,433, filed Jul. 2, 1999, now U.S. Pat. No. 6,463,474, the entire contents of which are hereby incorporated by reference as if fully set forth herein.

FIELD OF THE INVENTION

The present invention generally relates to management of computer networks, and relates more specifically to authentication and authorization mechanisms for network devices such as routers and firewalls.

BACKGROUND OF THE INVENTION

Computer users often access information, computer files, or other resources of computer networks from locations that are geographically or logically separate from the networks. This is referred to as remote access. For example, a user of a host or client that is part of a local area network ("LAN") may want to retrieve information that resides on a computer that is part of a remote network. Before a user can gain access to that computer, the user must first obtain permission to do so. In the interest of data integrity, and data confidentiality, many computer networks have implemented integrity and access control mechanisms to guard against unwanted network traffic or access by unauthorized users. On the other hand, a corporation may institute policies that restrict its employees from accessing certain web sites on the internet while using the corporation's computer resources. For example, Corporation C may disallow access to pornographic web sites. Corporation C's access control mechanism would prevent the employees from accessing such sites.

An example of an access control mechanism is a server that implements authentication, authorization, and accounting ("AAA") functions. Authentication is the process of verifying that the user who is attempting to gain access is authorized to access the network and is who he says he is. Generally, after authentication of a user, an authorization phase is carried out. Authorization is the process of defining what resources of the network an authenticated user can access.

Several authentication and authorization mechanisms are suitable for use with operating systems that are used by network devices, such as the Internetworking Operating System ("IOS") commercially available from Cisco Systems, Inc. However, most prior authentication and authorization mechanisms are associated with dial-up interfaces, which can create network security problems. In a dial-up configuration, a remote client uses a telephone line and modem to dial up a compatible modem that is coupled to a server of the network that the remote client wishes to access. In another dial-up configuration, a remote client first establishes a dial-up connection to a server associated with an Internet Service Provider, and that server then connects to the network server through the global, public, packet-switched internetwork known as the Internet. In this configuration, the network server is coupled directly or indirectly to the Internet.

Unfortunately, information requests and other traffic directed at a network server from the Internet is normally

considered risky, untrusted traffic. An organization that owns or operates a network server can protect itself from unauthorized users or from unwanted traffic from the Internet by using a firewall. A firewall may comprise a router that executes a "packet filter" computer program. The packet filter can selectively prevent information packets from passing through the router, on a path from one network to another. The packet filter can be configured to specify which packets are permitted to pass through the router and which should be blocked. By placing a firewall on each external network connection, an organization can prevent unauthorized users from interfering with the organization's network of computers. Similarly, the firewall can be configured to prevent the users of the organization's network of computers from accessing certain undesirable web sites on the Internet.

One common method of remote access using the Internet is telnet, a protocol used to support remote login sessions that defines how local and remote computers talk to each other to support a remote login session. "Telnet" is also the name of a remote login program commonly used in networks based on Transmission Control Protocol/Internet Protocol ("TCP/IP"), a set of protocols that define how communications occur over the Internet. Past authentication and authorization mechanisms were produced to work with firewalls in the context of telnet. An example of an authentication and authorization mechanism that works with telnet is "Lock and Key" for IOS, commercially available from Cisco Systems, Inc.

However, a major drawback of telnet is that the client must know, before making any connection request, the Internet Protocol address ("IP address") of the firewall that is protecting the target network which the client is attempting to access. An IP address is a unique 32-bit binary number assigned to each firewall, router, host computer or other network element that communicates using IP. Obtaining the IP address of a firewall can be inconvenient or impractical because there are so many IP addresses currently assigned to network devices. Further, IP addresses normally are guarded closely by the network owner, because knowledge of an IP address enables unauthorized traffic to reach the device identified by the IP address.

Moreover, once a user successfully uses the authentication and authorization mechanism to secure a logical path through the firewall, the user may be restricted to one type of network traffic for the connection. For example, a firewall can be configured to provide a path through the firewall for a specific type of network traffic as specified by a user profile that is associated with each authenticated user. The user profile contains information on what the user is authorized to do on the network. The user profile may specify, for example, that the user may use only File Transfer Protocol ("FTP") traffic. Thus, the user may use the path through the firewall only for FTP traffic, for the duration of that connection. Furthermore, the user profile associated with the user contains a specific IP address that specifies the host or client from which the user can attempt to secure a logical path through the firewall. Thus, a user is not free to use any one of several computers that may be available to access the target network. Also, the user may not be free to use a client in a network that employs Dynamic Host Configuration Protocol (DHCP). DHCP assigns dynamic IP addresses to the devices on a network. Thus, a client in a DHCP environment can have a different IP address every time it connects to the network.

Based on the foregoing, there is a clear need for a mechanism allowing users to use remote access via the Internet without requiring advance knowledge of the IP address of the firewall router, and without restricting a user to a particular host or client.

In particular, there is a need for an authentication and authorization mechanism in the context of remote access via the Internet that does not rely on telnet and that allows the passage of different types of traffic for a given connection.

SUMMARY OF THE INVENTION

The foregoing needs, and other needs and objects that will become apparent for the following description, are achieved in the present invention, which comprises, in one aspect, a method of controlling access of a client to a network resource using a network device that is logically interposed between the client and the network resource, the method comprising creating and storing client authorization information at the network device, wherein the client authorization information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource; receiving a request from the client to communicate with the network resource; determining, at the network device, whether the client is authorized to communicate with the network resource based on the authorization information; and reconfiguring the network device to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information.

One feature of this aspect is that creating and storing client authorization information comprises the steps of creating and storing in the network device a set of authorization information for each client that communicates with the network device. According to another feature of this aspect is that creating and storing client authorization information comprises the steps of creating and storing in the network device an authentication cache for each client that communicates with the network device. In another feature, creating and storing client authorization information comprises the steps of creating and storing in the network device a plurality of authentication caches, each authentication cache uniquely associated with one of a plurality of clients that communicate with the network device, each authentication cache comprising information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource

According to still another feature, determining whether the client is authorized to communicate with the network resource comprises the step of determining whether information in the request identifying the client matches information in a filtering mechanism of the network device and the authorization information stored in the network device.

In another feature, determining whether the client is authorized to communicate with the network resource comprises the steps of: determining whether a source IP address of the client in the request matches information in a filtering mechanism of the network device; and if so, determining whether the source IP address matches the authorization information stored in the network device.

In another feature, determining whether the client is authorized to communicate with the network resource comprises the steps of: determining whether a source IP address of the client in the request matches information in an a filtering mechanism of the network device; determining whether the source IP address matches the authorization information stored in the network device; and when the source IP address fails to match the authorization information stored in the network device, determining if user identifying information received from the client matches a profile associated with the user that is stored in an authentication server that is coupled to the network device.

In another feature, determining whether the client is authorized to communicate with the network resource comprises the steps of: determining whether client identifying information in the request matches information in a filtering mechanism of the network device; determining whether the client identifying information matches the authorization information stored in the network device; and only when the client identifying information fails to match the authorization information stored in the network device, then: creating and storing new authorization information in the network device that is uniquely associated with the client; requesting login information from the client; authenticating the login information by communicating with an authentication server that is coupled to the network device; and updating the new authorization information based on information received from the authentication server.

According to another feature, requesting login information from the client comprises sending a Hypertext Markup Language login form to the client to solicit a username and a user password; and authenticating the login information by communicating with an authentication server that is coupled to the network device comprises determining, from a profile associated with a user of the client stored in the authentication server, whether the username and password are valid.

In another feature, the method further comprises the steps of: creating and storing an inactivity timer for each authentication cache, wherein the inactivity timer expires when no communications are directed from the client to the network resource through the network device during a pre-determined period of time; removing the updated authentication information when the inactivity timer expires.

In another feature, determining whether the client is authorized to communicate with the network resource comprises the steps of: determining whether a source IP address in the request matches information in a filtering mechanism of the network device; determining whether the source IP address matches the authorization information stored in the network device; and only when the source IP address fails to match the authorization information stored in the network device, then: creating and storing in the network device a new authentication cache that is uniquely associated with the client; requesting login information from the client; authenticating the login information by communicating with an authentication server that is coupled to the network device; and updating the new authentication cache based on information received from the authentication server.

According to another feature, reconfiguring the network device comprises the steps of creating and storing one or more commands to the network device whereby one or more interfaces of the network device are modified to permit communications between the client and the network resource.

In another feature, the method further involves instructing the client to reload the network resource that was identified in the request from the client when it is determined that the client is authorized to communicate with the network resource.

According to another feature, the method further comprises the steps of waiting a pre-determined period of time, and instructing the client to reload the network resource that was identified in the request from the client when it is determined that the client is authorized to communicate with the network resource.

In another feature, the network device comprising a firewall that protects the network resource by selectively blocking messages initiated by client and directed to the network resource, the firewall comprising an external interface and an internal interface, the firewall comprising an Output Access Control List at the internal interface and an Input Access Control List at the external interface, wherein reconfiguring the network device comprises the step of: substituting the IP address in a user profile information associated with a user of the client to create a new user profile information, wherein the user profile associated with the user of the client is received from an authentication server that is coupled to the network device; and adding the new user profile information as temporary entries to the Input Access Control List at the external interface and to the Output Access Control List at the internal interface.

According to still another feature, the method further involves: creating and storing an inactivity timer for the authorization information, wherein the inactivity timer expires when no communications are directed from the client to the network resource through the network device during a pre-determined period of time; associating the temporary entries with the authorization information and the client; and removing the temporary entries and the authorization information from the network device if the inactivity timer expires.

In another feature, the authorization information includes a table of hashed entries and wherein associating the temporary entries to the authorization information further comprises storing the temporary entries in the table of hashed entries.

In another feature, the network device comprising a firewall that protects the network resource by selectively blocking messages initiated by client and directed to the network resource, the firewall comprising an external interface and an internal interface, the firewall comprising an Output Access Control List at the external interface and an Input Access Control List at the internal interface, wherein reconfiguring the network device comprises the step of: substituting the IP address in a user profile information associated with a user of the client to create a new user profile information, wherein the user profile associated with the user of the client is received from an authentication server that is coupled to the network device; and adding the new user profile information as temporary entries to the Input Access Control List at the internal interface and to the Output Access Control List at the external interface.

In another feature, the method further involves: creating and storing an inactivity timer for the authorization information, wherein the inactivity timer expires when no communications are directed from the client to the network resource through the network device during a predetermined period of time; associating the temporary entries with the authorization information and the client; and removing the temporary entries and the authorization information from the network device if the inactivity timer expires.

In another feature, the authorization information includes a table of hashed entries and wherein associating the temporary entries to the authorization information further comprises storing the temporary entries in the table of hashed entries.

According to another aspect, the invention encompasses computer system for controlling access of a client to a network resource using a network device that is logically interposed between the client and the network resource, comprising: one or more processors; a storage medium carrying one or more sequences of one or more instructions including instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of: creating and storing client authorization information at the network device, wherein the client authorization information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource; receiving a request from the client to communicate with the network resource; determining, at the network device, whether the client is authorized to communicate with the network resource based on the authorization information; and reconfiguring the network device to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information.

According to another aspect, the invention involves a router that is logically interposed between a client and a network resource and that controls access of the client to the network resource, comprising: one or more processors; a storage medium carrying one or more sequences of one or more instructions including instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of: creating and storing client authorization information at the router, wherein the client authentication information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource; receiving a request from the client to communicate with the network resource; determining, at the router, whether the client is authorized to communicate with the network resource based on the authorization information; and reconfiguring the router to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information.

In other aspects, the invention encompasses a computer apparatus, a computer readable medium, and a carrier wave configured to carry out the foregoing steps.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

FIG. **1** is a block diagram that illustrates a computer system upon which an embodiment may be implemented;

FIG. **2** is a block diagram of a system providing an authentication proxy in a network environment;

FIG. **3** is a block diagram of the system in FIG. **2** showing certain internal details;

FIG. **4** is a block diagram of the system in FIG. **3** showing certain paths of network traffic;

FIG. **5**A illustrates a display of a graphical user interface containing a dialog box for soliciting a username and password;

FIG. **5**B illustrates a display of the graphical user interface informing of an authentication success;

FIG. **6** is a state diagram of states in which an authentication cache may execute;

FIG. 7A is a flow diagram of a process of proxy authentication;

FIG. 7B is a flow diagram of further steps in the process of FIG. 7A.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A method and apparatus for authentication and authorization proxy mechanisms for firewalls that protect networks is described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

Operational Context

The present invention may be implemented using various client protocols such as Telnet, File Transfer Protocol (FTP), or HyperText Transfer Protocol (HTTP). For purposes of illustration, the invention is described in the context of an HTTP client protocol.

In one embodiment, a user of a client that is part of a local area network ("LAN") attempts to remotely access a server ("target server") or some other resource, such as a peer client or device. The target server and or peer are part of a packet-switched private network that operates using TCP/IP and other Internet standards ("intranet"). The client is connected to the Internet through the LAN, and the intranet is also connected to the Internet. Alternatively, the client may be a stand-alone computer connected to the Internet through a dial-up connection or a digital communication service such as an Integrated Services Digital Network (ISDN) connection. In another embodiment, a user of a client from within the intranet attempts to access a target server or other resource that is not part of the same intranet as that of the client.

When the target server executes an HTTP server, the client can remotely access the target server over the Internet by using a Web browser to specify a Web page on the target server. Using a Web browser to specify a Web page is hereafter referred to as an "HTTP request" or as "transmitting HTTP packets." A Web page of the target server may be accessed using identifying information, such as a Uniform Resource Locator ("URL") and therefore the Web page is sometimes called the "target URL."

The HTTP packets are intercepted by a firewall that protects the intranet from unwanted network traffic originating from the Internet (inbound traffic) and can prevent users of clients from within the intranet from accessing undesirable web sites on the Internet (outbound traffic). For purposes of illustration, use of an embodiment with inbound traffic is described in further detail below.

Upon intercepting the HTTP packets, the firewall requests, from the client, authentication information such as username and password. In response to receiving the authentication information, the firewall performs an authentication and authorization process. If the username is successfully authenticated, then the firewall is dynamically configured to open a passageway for the HTTP packets as well as other types of network traffic initiated from the user on the client. The other types of network traffic that are permitted through the passageway are specified in a user profile for that particular user. In this context, "open a passageway" means that the firewall re-configures itself, in response to successful authentication, so that packets that would otherwise be barred are now allowed to pass.

In this configuration, the firewall provides an authentication and authorization mechanism that substitutes for an authentication and authorization mechanism elsewhere in the network. Accordingly, the mechanism described in this document is referred to as an "Authentication Proxy." The Authentication Proxy may comprise one or more software components, executed by a router. In one embodiment, the Authentication Proxy can be enabled on a router interface to intercept traffic initiated from a client that is not yet authenticated. The Authentication Proxy is responsible for validating the user associated with the client and for applying the appropriate user profile to the router interface. The authentication and authorization process and the dynamic configuration of the firewall are described in further detail below.

FIG. **2** is a block diagram of a system **200** in which an embodiment of an Authentication Proxy can be used. Generally, system **200** includes a LAN **206**, and a local, packet-switched network that uses Internet protocols, or intranet, **216**. The LAN **206** and the intranet are both connected to a global network such as the Internet. The LAN **206** and intranet **216** are respectively located in logically distinct regions, such as first region **202** and second region **204**, which may be geographically separate. A firewall router **210** is logically interposed between LAN **206** and the intranet **216**.

LAN **206** is a local area network comprising any number of network devices **208***a*, **208***b*, **208***c* interconnected by one or more communications channels **209**. Ethernet, Token Ring, other protocols can characterize the communications channels **209**.

Firewall router **210** is a specialized router that carries out firewall functions. The firewall router **210** is coupled to intranet **216**, and an authentication and authorization server **218** ("AAA server"). The firewall router **210** controls remote access to intranet **216**. AAA server **218** is a computer, or a group of hardware or software components or processes that cooperate or execute in one or more computer systems. The AAA server **218** has access to a database **220** that stores authentication and authorization information on users ("user profile"). The firewall router **210** cooperates with the AAA server **218** to perform authentication and authorization services. In this way, firewall router **210** restricts and controls access of traffic originating from intranet **216** and directed to LAN **206**.

Intranet **216** is one or more interconnected networks ("internetwork") each of which may comprise any number of network devices. A target server **222** is part of Intranet **216** and is a computer system that may be remotely accessed by a client on LAN **206**. In certain embodiments, the AAA server **218** and database **220** may be coupled indirectly to firewall router **210** through the Internet. In this configuration, the AAA server **218** and database **220** are said to be one or more "hops" removed from the firewall router **210**. A hop is the next place in the network to send a packet so that a packet will eventually reach its destination.

FIG. **3** is a block diagram showing certain internal details of the system in FIG. **2**. In this example, one of the network devices **208***a*–**208***c* of LAN **206** is a client **306**, such as a personal computer or workstation. Client **306** is associated with a user **302**. In certain embodiments, client **306** is configured with or coupled to multiple modems or ISDN bearer channels that can be used to establish one or more connections **308, 310** from client **306** to firewall router **210**. In one embodiment, client **306** runs a browser **304**, which is an application program used to retrieve and display Web pages or other information stored in target server **222**. Commercial examples of browser **304** include Netscape Navigator® or Microsoft Internet Explorer®. User **302** can use browser **304** to send an HTTP request over LAN **206** and Intranet **216** to target server **222**.

Authentication and Authorization

FIG. **4** is a block diagram of the system of FIG. **3** that illustrates providing an Authentication Proxy.

FIG. **7A** and FIG. **7B** are flow diagrams that illustrate a method for carrying out proxy authentication at a router. As an example, the method of FIG. 7A and FIG. 7B is described below in the context of the system of FIG. **4**. However, the method is not limited to this context.

As in FIG. **3**, the system of FIG. **4** includes User **302** who is associated with Client **306**. A Browser **304** is executed by Client **306**, which is part of LAN **206**. Client **206** communicates with firewall router **210** over LAN **206** using messages illustrated by paths **401**, **403**, **404**, **408***a*, **408***b*. Firewall router **210** is coupled to client **306**, to AAA Server **218**, and to intranet **216**. One function of Client **306** is to request and retrieve electronic documents, applications or services that are available at target server **222**.

A filtering mechanism **219** is part of the configuration of Authentication Proxy **400**. The filtering mechanism **219** contains information identifying one or more IP addresses of clients that are authenticated in the network to which the firewall **210** belongs.

Paths of network traffic are depicted by solid lines or dotted lines with arrowheads. Paths **401, 403, 404, 408***a*, **408***b* illustrate network traffic communicated between client **306** and firewall router **210**. Paths **405** and **406** illustrate network traffic communicated between firewall router **210** and AAA server **218**. Paths **409** and **410** illustrate network traffic communicated between the client **306** and target server **222**. Each path represents one or more packets, messages or sessions communicated among the respective end elements. Paths **409** and **410** pass through firewall router **210** but do not stop within the firewall router.

Firewall router **210** has an external interface **420** and an internal interface **422**. Each interface **420, 422** defines how firewall router **210** regulates the flow of traffic arriving at or sent from the respective interface. The external interface **420** includes an input Access Control List ("ACL") **424**, and an output ACL **426**. Internal interface **422** also includes an input ACL **428** and an output ACL **430**.

Access control lists filter packets and can prevent certain packets from entering or exiting a network. Each ACL is a list of information that firewall router **210** may use to determine whether packets arriving at or sent from a particular interface may be communicated within or outside the firewall router. For example, in an embodiment, input ACL **424** may comprise a list of IP addresses and types of allowable client protocols. Assume that firewall router **210** receives an inbound packet from client **306** at external interface **420** that is intended for target server **222**. If the IP address of client **306** is not stored in input ACL **424**, then firewall router **210** will not forward the packet further within the circuitry or software of the firewall router. Output ACL **426** similarly controls the delivery of packets from firewall router **210** to resources located outside external interface **420**. Input ACL **428** and output ACL **430** govern packet flow to or from internal interface **422**.

The firewall router **210** also includes any number of authentication caches **432, 434**. The access control lists are linked to the authentication caches. Each authentication cache represents a valid user authentication. Each authentication cache may include a table of hashed entries of information such as a source IP address, a destination IP address, a source port value, a destination port value, and state information.

Firewall router **210** also includes Authentication Proxy **400**. In one embodiment, Authentication Proxy **400** is one or more software elements, modules or processes executed by firewall router **210** and coupled to the external interface **420**, internal interface **422**, and authentication caches **432, 434**. Authentication Proxy **400** may be configured to carry out the functions described in this document.

Authentication

In an embodiment, Authentication Proxy **400** is activated or enabled at the firewall router **210**. Authentication Proxy **400** may be implemented as a software process within firewall router **210** that may be accessed using commands in a standard command-line router programming language.

For purposes of illustrating an example of operation of authentication proxy **400**, assume that Authentication Proxy **400** receives a request for something in a network that is protected by the Authentication Proxy, as shown by block **702** of FIG. **7A**. For example, User **302** uses browser **304** to send an HTTP request from client **306** for an electronic document, application or resource available at target server **222**. User **302** is not authenticated in intranet **216**. The HTTP request travels along path **401** from client **306** to firewall router **210**. Each packet of an HTTP request includes a header portion that contains one or more fields of information. The fields include, among other things, values for source IP address and destination IP address of that packet.

In block **704**, packets of the request are examined. For example, when the HTTP request arrives at the external interface **420** of the firewall router **210**, Authentication Proxy **400** examines packets of the request. In block **706**, the process determines whether a source IP address of the request is found in the standard access control list. For example, Authentication Proxy **400** determines whether the source IP address in the header field of the packets corresponds to any entry in the filtering mechanism **219** configured in the Authentication Proxy **400**.

If the test of block **706** is affirmative, then control passes to block **708** in which the authentication caches are searched for the source IP address. In block **710**, the process tests whether the source IP address is found. For example, if Authentication Proxy **400** determines that the source IP address matches at least one IP address stored in the filtering mechanism **219**, then the Authentication Proxy **400** attempts to authenticate the user **302**. In the preferred embodiment, Authentication Proxy **400** searches authentication caches **432, 434** for the source IP address. The goal of this search is to determine if the source IP address of the HTTP packet corresponds to an entry in any of the authentication caches **432, 434**.

Assume that the filtering mechanism **219** contains a source IP address that matches the IP address of client **306**, so that the test of block **706** is affirmative. However, User **302** is not yet authenticated, so that the test of block **710** is negative. Thus, the authentication caches have no hashed entries that match the source IP address found in the header field of the HTTP packet. As a result, without further action the firewall router **210** will intercept and will not forward the

HTTP packet, instead of providing it with a logical passageway through the router.

In one embodiment, at this stage, control is passed to block **720** of FIG. **7B**. Preferably, a new authentication cache **436** is created for User **302**, as shown in block **720**. Each authentication cache may operate in a plurality of states. FIG. **6** is a state diagram that illustrates the states in which an authentication cache may operate. Initially, the state of the new authentication cache **436** is set to the HTTP_Init state **602**, as shown by block **722**. The HTTP_Init state **602** indicates that the authentication and authorization state for User **302** is at an initial state. In one embodiment, an HTTP packet is provided with a logical passageway through the firewall router **210** only when the state of an authentication cache that contains information about User **302** is set to the HTTP_Estab state **606**, which is described further below.

If the source IP address of the HTTP packet from client **306** does not match any of the entries in the filtering mechanism **219**, then Authentication Proxy **400** denies passage to the HTTP packet and makes no attempt at authentication, as shown by block **707** of FIG. **7A**. As a result, advantageously, the packet is turned away at the interface and never reaches internal software and hardware elements of the firewall router.

If Authentication Proxy **400** is not configured with the filtering mechanism **219**, then Authentication Proxy **400** will intercept traffic, for purposes of authentication, from all hosts whose connection initiating packets arrive at a firewall interface for which Authentication Proxy **400** has been enabled. It is not practical or desirable to attempt authentication for every packet that arrives at the firewall. For example, there is no point in authenticating packets that arrive from sites known to be hostile or sites of an unknown nature. Thus, in one embodiment, a filtering mechanism is defined to enable the Authentication Proxy **400** to bypass traffic from such sites.

Referring again to FIG. **7B**, after the new authentication cache is created, login information is requested from the client, as shown in block **724**. For example, Authentication Proxy **400** obtains authentication information from User **302** by sending a login form to client **306**. The login form is an electronic document that requests User **302** to enter username and password information, as shown by path **403**.

FIG. **5A** is an example of a graphical user interface ("GUI") representation of the login form **500** that may be sent by Authentication Proxy **400**. The form **500** may be displayed by browser **304**, as indicated by command area **502**. Form **500** also includes a data entry pane **503** having a Username field **504**, and a Password field **506**. A User **302** may enter username information in Username field **504** and may enter password information in Password field **506**. To communicate the username and password information to Authentication Proxy **400**, the user selects a "Submit" button **508**. In response, the login information is communicated to firewall router **210**, as indicated by path **404**.

The login form may be composed using the HyperText Markup Language ("HTML") format. Table 1 presents source code for an appropriate HTML document that may be used for the login form of FIG. **5A**.

TABLE 1

EXAMPLE LOGIN FORM

```
<HTML>
  <HEAD>
   <TITLE>
     Authentication Proxy Login Page
   </TITLE>
  </HEAD>
  <BODY BGCOLOR=#FFFFFF>
   <H1>Cisco "router name" Firewall<H1>
   <H2>Authentication Proxy<H2><BR><BR><P><P><P>
   <FORM METHOD=POST ACTION=\"\">
   <INPUT TYPE=HIDDEN NAME=TIMETAG VALUE=>
    Username: <INPUT TYPE=TEXT NAME=UNAME><BR><BR>
    Password: <INPUT TYPE=PASSWORD NAME=PWD><BR><BR>
   <INPUT TYPE=SUBMIT NAME=SUBMIT VALUE=SUBMIT>
   </FORM>
  </BODY>
</HTML>
```

Login information is received from the client, as shown by block **726**. Block **726** may involve receiving username and password information in an HTTP message that is generated when a user fills in and submits the form of Table 1. In block **728**, the user is authenticated using the login information and the AAA server **218**. For example, upon receipt of the username and password information, Authentication Proxy **400** attempts to authenticate the user by sending the username and password to the AAA server **218**, as shown by path **405**. AAA server **218** has access to database **220**, which contains user profiles of authorized users.

In block **730**, the process tests whether authentication is successful. Successful authentication will occur when AAA server **218** verifies that the username and password information are recognized and correct. If Authentication Proxy **400** successfully authenticates User **302** using AAA Server **218**, then AAA server **218** informs firewall router **210** of the successful authentication, as shown by path **406**. In block **732**, the process updates the current authentication cache with the source IP address of the client and with information contained in the user profile of User **302**. In block **734**, the firewall is re-configured to allow packets using protocols specified in user's profile of User **302** and associated with that IP address to pass through the firewall freely. Block **734** may involve creating and executing one or more router commands. In this way, subsequent authentication attempts against the authentication cache at the firewall will succeed, precluding the need to connect to AAA Server **218** repeatedly. This provides a significant advantage and improvement in authentication speed and efficiency.

Authentication Proxy **400** then informs Client **306** that authentication succeeded. As shown in block **736**, the client is notified. In the preferred embodiment, Authentication Proxy **400** sends an HTML page containing a success message, as shown by path **408***a*.

FIG. **5B** is an example of an HTML page that Authentication Proxy **400** may send to Client **306**. Pane **522** of window **520** contains an "Authentication Successful" message **524**.

Pane **522** also includes a second message **526** informing Client **306** that further action is being taken. For example, the second message **526** may inform Client **306** that the resource originally requested from target server **222** is now being loaded or accessed. In the preferred embodiment, the original HTTP request is automatically forwarded by firewall router **210** to target server **222** without further intervention by Client **306** or User **302**. Thus, Authentication Proxy **400** automatically interrupts and resumes the request.

As shown in block **738**, the process waits. For example, after Authentication Proxy **400** sends the "Authentication Success" message **524** to User **302**, the Authentication Proxy enters a wait state for a short, pre-determined period of time. During the wait state, a short period of time is allowed to elapse to enable client **306** and firewall router **210** to communicate handshaking messages and carry out related processing associated with establishing an HTTP connection. The delay period also allows the firewall router enough time to execute any commands that are issued as part of block **734**. In one embodiment, a period of three (3) seconds elapses.

In block **740**, the process sends a page reload instruction to the client. For example, after the delay period, Authentication Proxy **400** sends one or more messages to Client **306** that instruct the client to reload the target URL, as indicated by path **408***b* of FIG. **4**. In response, Client **306** re-issues the original HTTP request for a resource on target server **222**. By the time that the re-issued HTTP request arrives from Client **306** at firewall router **210**, the firewall router has been re-configured to provide an logical path **409** circumscribed only by the user profile of User **302** for the connection to the target server using the process described further below. Similarly, return traffic may travel from target server **222** to Client **306** on path **410** subject to the authorization information of the user profile of User **302**.

If the authentication is not successful, as shown in block **736** and block **738**, the process may notify the client with an appropriate message or page, and block traffic from the source IP address. Alternatively, Authentication Proxy **400** may permit Client **306** to re-try authentication a predetermined number of times.

Creating a Path Through the Firewall Using Dynamic Access Control List

When User **302** is authenticated successfully, Authentication Proxy **400** causes a user profile of User **302** to be downloaded from the AAA server **218** to the firewall router **210**. Authentication Proxy **400** uses the user profile to dynamically configure the external interface **420** and internal interface **422** of firewall router **210** to provide a passageway for network traffic that initiates from User **302** on client **306**. In this way, the types of traffic as specified by the user profile of User **302** and initiating from User **302** in the current session will pass freely through firewall router **210** to its destination.

In one embodiment, an authorized passageway through firewall router **210** is created using dynamic access control lists. In this embodiment, one or more entries may be added dynamically to each access control list **424, 426, 428, 430** of external interface **420** and internal interface **422**.

When Authentication Proxy **400** sends a successful login confirmation message to User **302**, a user profile of User **302** is downloaded from AAA server **218**. In the preferred embodiment, the user profile is stored and retrieved in the form of one or more text commands, called proxy-access-list commands. Authentication Proxy **400** parses the proxy-access-list commands. Table 2 below contains samples of proxy-access-list commands. As each command is parsed, Authentication Proxy **400** replaces the source IP address field of the command with the IP address of the client **306**, to result in a modified proxy-access-list command. Each modified proxy-access-list command is added as a temporary entry to the access control lists at the external interface **420** and internal interface **422**. Specifically, proxy-access-list commands are added as temporary entries to the input ACL **424** of the router's external interface **420** and to the output ACL **430** of the router's internal interface **422**. Thus, access control lists **424, 426, 428, 430** will grow and shrink dynamically as entries are added and deleted.

When the temporary entries have been added to the ACL at the appropriate interfaces of the firewall router **210**, the state of the current authentication cache is set to the HTTP_FINWAIT state **604**. The HTTP_FINWAIT state **604** indicates that the authentication cache is associated with an authenticated user, but that its associated client is not yet connected to a target server.

TABLE 2

EXAMPLE PROXY-ACCESS-LIST COMMANDS
permit tcp host 192.168.25.215 any eq telnet
permit tcp host 192.168.25.215 any eq ftp
permit tcp host 192.168.25.215 any eq ftp-data
permit tcp host 192.168.25.215 any eq smtp
deny tcp any any eq telnet
deny udp any any
permit tcp any any (76 matches)
permit ip any any

Authentication Cache Inactivity Timers

Each authentication cache may have an inactivity timer. In the preferred embodiment, an inactivity cache is a software process that is maintained for each authentication cache based on the amount of traffic coming through firewall router **210** from Client **306**. If the amount of traffic through the router for a particular client falls below a pre-determined threshold over a pre-determined period of time, then Authentication Proxy **400** generates an idle timeout event.

When an idle timeout event is generated for a particular client, Authentication Proxy **400** deletes the authentication cache associated with that client, and deletes all temporary entries in the access control lists that are associated with that particular client. This approach saves memory and ensures that the ACLs are regularly pruned.

For example, assume that the pre-determined inactivity timer value is 60 minutes. If Client **306** does not initiate any network traffic through the firewall router **210** for 60 minutes or more, then any subsequent traffic initiated from client **306** will be denied passage through the firewall router **210**. In one embodiment, Authentication Proxy **400** will prompt User **302** to renew authentication for a new HTTP connection.

Preferably, the temporary entries in the ACLs are not automatically deleted when a user terminates a session. Instead, the temporary entries remain stored until the configured timeout is reached, or until they are specifically deleted by the system administrator. Using this process, the user is not required to log in a second time in the event of an inadvertent or transient disconnection, but unused entries are removed from the ACLs when a true idle condition occurs.

In addition, each temporary entry to an ACL **424, 426, 428, 430** is linked to an associated authentication cache **432, 434**. Preferably, a temporary entry of an ACL is linked to an associated authentication cache by hashing the temporary entries and storing the hashed entries in the hash table that is maintained by the authentication cache.

Connection and Communication with Target Server

When the interfaces of firewall router **210** have been configured with the new temporary entries in the ACLs, the result is that a logical passageway is opened through the firewall to allow certain types of traffic specified in the user profile of User **302** and initiating from Client **306** to pass unobstructed to target server **222**. Using standard HTTP request-response communications, Client **306** establishes an IITTP connection to target server **222**. When the connection to target server **222** is established, Authentication Proxy **400** changes the state of the authentication cache associated with the User **302** to the HTTP_ESTAB state **606**.

As long as an idle timeout event does not occur, an authentication cache associated with a client remains valid. As long as the authentication cache associated with a client remains valid, the firewall router **210** does not attempt to intercept any subsequent traffic initiating from that client.

For example, assume that Client **306** continues to initiate network traffic or packets for passage through the firewall router **210** at least once every 60 minutes. Thus, the authentication cache **436** associated with client **306** remains valid and is not removed. Corresponding entries in the ACLs **424**, **426**, **428**, **430** also remain valid. When a packet arrives at the firewall router **210**, Authentication Proxy **400** determines that the source IP address of the packet matches an entry in the authentication cache **436**. Accordingly, Authentication Proxy **400** will allow the packet to pass through firewall router **210**.

Moreover, subsequent traffic initiating from client **306** may travel to any destination, as specified by the temporary entries to the dynamic ACLs that are associated with client **306**. When a packet from client **306** arrives at firewall router **210**, Authentication Proxy **400** checks the hashed entries in the authentication cache **436** for a match in the destination IP address. If a match is found, then the Authentication Proxy **400** will allow the packet to pass through firewall router **210** to the specified destination.

In addition, the temporary entries to the dynamic ACLs **424**, **426**, **428**, **430** may specify various permissible communication protocols that client **306** may use. Thus, client **306** may initiate network traffic using any communication protocol specified in its associated temporary ACL entries, as long as an idle timeout event has not occurred. The client **306** is not restricted to TCP/IP protocol that was used to send the initial HTTP request. Examples of other communication protocols that client **306** may use include telnet, Internet Control Message Protocol ("ICMP"), and File Transfer Protocol ("FTP").

Hardware Overview

FIG. **1** is a block diagram that illustrates a computer system **100** upon which an embodiment of the invention may be implemented. In one embodiment, computer system **100** is a firewall device, such as a router.

Computer system **100** includes a bus **102** or other communication mechanism for communicating information, and a processor **104** coupled with bus **102** for processing information. Computer system **100** also includes a main memory **106**, such as a random access memory (RAM) or other dynamic storage device, coupled to bus **102** for storing information and instructions to be executed by processor **104**. Main memory **106** also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor **104**. Computer system **100** further includes a read only memory (ROM) **108** or other static storage device coupled to bus **102** for storing static information and instructions for processor **104**. A storage device **110**, such as non-volatile random-access memory (NVRAM), is provided and coupled to bus **102** for storing information and instructions.

Computer system **100** may be coupled via communication interface **117** to a terminal **112**, such as a cathode ray tube (CRT) dumb terminal or workstation, for receiving command-line instructions from and displaying information to a computer user. Terminal **112** includes an input device such as a keyboard, and may include a cursor control such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor **104**.

Computer system **100** has a switching system **116** which provides a plurality of links or interfaces to a network. Switching system **116** provides a way to connect an incoming network link **114** to an outgoing network link **118**. There may be many links **114**, **118**.

The invention is related to the use of computer system **100** for regulating packet traffic in an integrated services network. According to one embodiment of the invention, regulating packet traffic in an integrated services network is provided by computer system **100** in response to processor **104** executing one or more sequences of one or more instructions contained in main memory **106**. Such instructions may be read into main memory **106** from another computer-readable medium, such as storage device **110**. Execution of the sequences of instructions contained in main memory **106** causes processor **104** to perform the process steps described herein. In alternative embodiments, hardwired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor **104** for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, NVRAM, such as storage device **110**, or magnetic or optical disks. Volatile media includes dynamic memory, such as main memory **106**. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus **102**. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor **104** for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system **100** can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus **102**. Bus **102** carries the data to main memory **106**, from which processor **104** retrieves and executes the instructions. The instructions received by main memory **106** may optionally be stored on storage device **110** either before or after execution by processor **104**.

Computer system **100** also includes a communication interface **117** coupled to bus **102**. Communication interface **117** provides a two-way data communication coupling to a network link **120** that is connected to a local network **122**. For example, communication interface **117** may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface **117** may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface **117** sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link **120** typically provides data communication through one or more networks to other data devices. For example, network link **120** may provide a connection through local network **122** to a host computer **124** or to data equipment operated by an Internet Service Provider (ISP) **126**. ISP **126** in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" **128**. Local network **122** and Internet **128** both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link **120** and through communication interface **117**, which carry the digital data to and from computer system **100**, are exemplary forms of carrier waves transporting the information.

Computer system **100** can send messages and receive data, including program code, through the network(s), network link **120** and communication interface **117**. In the Internet example, a server **130** might transmit a requested code for an application program through Internet **128**, ISP **126**, local network **122** and communication interface **117**. In accordance with the invention, one such downloaded application provides for regulating packet traffic in an integrated services network as described herein.

The received code may be executed by processor **104** as it is received, and/or stored in storage device **110**, or other non-volatile storage for later execution. In this manner, computer system **100** may obtain application code in the form of a carrier wave.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

What is claimed is:

1. A computer-readable medium carrying one or more sequences of one or more instructions for controlling access of a client to a network resource using a network firewall routing device, the one or more sequences of one or more instructions including instructions which, when executed by one or more processors, cause the one or more processors to perform the steps of:

creating and storing client authorization information at the network firewall routing device that is logically interposed between the client and the network resource, wherein the client authorization information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client has with respect to the network resource;

receiving a request from the client to communicate with the network resource;

determining, at the network firewall routing device, whether the client is authorized to communicate with the network resource based on the authorization information; and

reconfiguring the network firewall routing device to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information.

2. A computer-readable medium as recited in claim **1**, wherein creating and storing client authorization information comprises the steps of creating and storing in a cache in the network routing device a set of authorization information for each client that communicates with the network routing device.

3. A computer-readable medium as recited in claim **1**, wherein creating and storing client authorization information comprises the steps of creating and storing in the network routing device an authentication cache for each client that communicates with the network routing device.

4. A computer-readable medium as recited in claim **1**, wherein creating and storing client authorization information comprises the steps of creating and storing in the network routing device a plurality of authentication caches, each authentication cache uniquely associated with one of a plurality of clients that communicate with the network routing device, each authentication cache comprising information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource.

5. A computer-readable medium as recited in claim **1**, wherein determining whether the client is authorized to communicate with the network resource comprises the step of determining whether information in the request identifying the client matches information in a filtering mechanism of the network routing device and the authorization information stored in the network routing device.

6. A computer-readable medium as recited in claim **1**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address of the client in a data packet of the request matches information in a filtering mechanism of the network routing device; and

if so, determining whether the source IP address matches the authorization information stored in the network routing device.

7. A computer-readable medium as recited in claim **1**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address of the client in a data packet of the request matches information in an a filtering mechanism of the network routing device;

if a match is found using the filtering mechanism, determining whether the source IP address matches the authorization information stored in the network routing device; and

when the source IP address fails to match the authorization information stored in the network routing device, determining if user identifying information received from the client matches a profile associated with the user that is stored in an authentication server that is coupled to the network routing device.

8. A computer-readable medium as recited in claim 1, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether client identifying information in the request matches information in a filtering mechanism of the network routing device;

if a match is found using the filtering mechanism, determining whether the client identifying information matches the authorization information stored in the network routing device; and

only when the client identifying information fails to match the authorization information stored in the network routing device, then:

creating and storing new authorization information in the network device that is uniquely associated with the client;

requesting login information from the client;

authenticating the login information by communicating with an authentication server that is coupled to the network routing device; and

updating the new authorization information based on information received from the authentication server.

9. A computer-readable medium as recited in claim 8, wherein:

requesting login information from the client comprises sending a Hypertext Markup Language login form from the network routing device to the client to solicit a username and a user password; and

authenticating the login information by communicating with an authentication server that is coupled to the network routing device comprises determining, from a profile associated with a user of the client stored in the authentication server, whether the username and password are valid.

10. A computer-readable medium as recited in claim 1, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address in the request matches information in a filtering mechanism of the network routing device;

determining whether the source IP address matches the authorization information stored in the network routing device using an authentication cache in the network routing device; and

only when the source IP address fails to match the authorization information stored in the network routing device, then:

creating and storing a new entry in the authentication cache that is uniquely associated with the client;

requesting login information from the client;

authenticating the login information by communicating with an authentication server that is coupled to the network routing device; and

updating the new entry in the authentication cache based on information received from the authentication server.

11. A computer-readable medium as recited in claim 1, wherein reconfiguring the network routing device comprises the steps of creating and storing one or more commands to the network routing device which, when executed by the network routing device, result in modifying one or more routing interfaces of the network routing device to permit communications between the client and the network resource.

12. A computer system for controlling access of a client to a network resource using a network firewall routing device, comprising:

one or more processor;

a storage medium carrying one or more sequences of one or more instructions including instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of:

creating and storing client authorization information at the network firewall routing device that is logically interposed between the client and the network resource, wherein the client authorization information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client has with respect to the network resource;

receiving a request from the client to communicate with the network resource;

determining, at the network firewall routing device, whether the client is authorized to communicate with the network resource based on the authorization information;

reconfiguring the network firewall routing device to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information;

wherein creating and storing client authorization information comprises the steps of creating and storing in a cache in the network routing device a set of authorization information for each client that communicates with the network routing device.

13. A computer system as recited in claim 12, wherein creating and storing client authorization information comprises the steps of creating and storing in the network routing device a plurality of authentication caches, each authentication cache uniquely associated with one of a plurality of clients that communicate with the network routing device, each authentication cache comprising information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource.

14. A computer system as recited in claim 12, wherein determining whether the client is authorized to communicate with the network resource comprises the step of determining whether information in the request identifying the client matches information in a filtering mechanism of the network routing device and the authorization information stored in the network routing device.

15. A computer system as recited in claim 12, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address of the client in a data packet of the request matches information in a filtering mechanism of the network routing device; and

if so, determining whether the source IP address matches the authorization information stored in the network routing device.

16. A computer system as recited in claim 12, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address of the client in a data packet of the request matches information in an a filtering mechanism of the network routing device;

if a match is found using the filtering mechanism, determining whether the source IP address matches the authorization information stored in the network routing device; and

when the source IP address fails to match the authorization information stored in the network routing device, determining if user identifying information received from the client matches a profile associated with the user that is stored in an authentication server that is coupled to the network routing device.

17. A computer system as recited in claim **12**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether client identifying information in the request matches information in a filtering mechanism of the network routing device;

if a match is found using the filtering mechanism, determining whether the client identifying information matches the authorization information stored in the network routing device; and

only when the client identifying information fails to match the authorization information stored in the network routing device, then:

creating and storing new authorization information in the network device that is uniquely associated with the client;

requesting login information from the client;

authenticating the login information by communicating with an authentication server that is coupled to the network touting device; and

updating the new authorization information based on information received from the authentication server.

18. A computer system as recited in claim **17**, wherein:

requesting login information from the client comprises sending a Hypertext Markup Language login form from the network routing device to the client to solicit a username and a user password; and

authenticating the login information by communicating with an authentication server that is coupled to the network routing device comprises determining, from a profile associated with a user of the client stored in the authentication server, whether the username and password are valid.

19. A computer system as recited in claim **12**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address in the request matches information in a filtering mechanism of the network routing device;

determining whether the source IP address matches the authorization information stored in the network routing device using an authentication cache in the network routing device; and

only when the source IP address fails to match the authorization information stored in the network routing device, then:

creating and storing a new entry in the authentication cache that is uniquely associated with the client;

requesting login information from the client;

authenticating the login information by communicating with an authentication server that is coupled to the network routing device; and

updating the new entry in the authentication cache based on information received from the authentication server.

20. A computer system as recited in claim **12**, wherein reconfiguring the network routing device comprises the steps of creating and storing one or more commands to the network routing device which, when executed by the network routing device, result in modifying one or more routing interfaces of the network routing device to permit communications between the client and the network resource.

21. A data packet firewall router that is logically interposed between a client and a network resource and that controls access of the client to the network resource, comprising:

one or more processors;

a storage medium carrying one or more sequences of one or more instructions including instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of:

creating and storing client authorization information at the router, wherein the client authentication information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client has with respect to the network resource;

receiving a request from the client to communicate with the network resource;

determining, at the router, whether the client is authorized to communicate with the network resource based on the authorization information;

reconfiguring the router to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information;

wherein creating and storing client authorization information comprises the steps of creating and storing in a cache in the network routing device a set of authorization information for each client that communicates with the network routing device.

22. A data packet router as recited in claim **21**, wherein creating and storing client authorization information comprises the steps of creating and storing in the network routing device a plurality of authentication caches, each authentication cache uniquely associated with one of a plurality of clients that communicate with the network routing device, each authentication cache comprising information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource.

23. A data packet router as recited in claim **21**, wherein determining whether the client is authorized to communicate with the network resource comprises the step of determining whether information in the request identifying the client matches information in a filtering mechanism of the network routing device and the authorization information stored in the network routing device.

24. A data packet router as recited in claim **21**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address of the client in a data packet of the request matches information in a filtering mechanism of the network routing device; and

if so, determining whether the source IP address matches the authorization information stored in the network routing device.

25. A data packet router as recited in claim **21**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address of the client in a data packet of the request matches information in an a filtering mechanism of the network routing device;

- if a match is found using the filtering mechanism, determining whether the source IP address matches the authorization information stored in the network routing device; and
- when the source IP address fails to match the authorization information stored in the network routing device, determining if user identifying information received from the client matches a profile associated with the user that is stored in an authentication server that is coupled to the network routing device.

26. A data packet router as recited in claim **21**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

- determining whether client identifying information in the request matches information in a filtering mechanism of the network routing device;
- if a match is found using the filtering mechanism, determining whether the client identifying information matches the authorization information stored in the network routing device; and
- only when the client identifying information fails to match the authorization information stored in the network routing device, then:
  - creating and storing new authorization information in the network device that is uniquely associated with the client;
  - requesting login information from the client;
  - authenticating the login information by communicating with an authentication server that is coupled to the network routing device; and
  - updating the new authorization information based on information received from the authentication server.

27. A data packet router as recited in claim **26**, wherein:

- requesting login information from the client comprises sending a Hypertext Markup Language login form from the network routing device to the client to solicit a username and a user password; and
- authenticating the login information by communicating with an authentication server that is coupled to the network routing device comprises determining, from a profile associated with a user of the client stored in the authentication server, whether the username and password are valid.

28. A data packet router as recited in claim **21**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

- determining whether a source IP address in the request matches information in a filtering mechanism of the network routing device;
- determining whether the source IP address matches the authorization information stored in the network routing device using an authentication cache in the network routing device; and
- only when the source IP address fails to match the authorization information stored in the network routing device, then:
  - creating and storing a new entry in the authentication cache that is uniquely associated with the client;
  - requesting login information from the client;
  - authenticating the login information by communicating with an authentication server that is coupled to the network routing device; and
  - updating the new entry in the authentication cache based on information received from the authentication server.

29. A data packet router as recited in claim **21**, wherein reconfiguring the network routing device comprises the steps of creating and storing one or more commands to the network routing device which, when executed by the network routing device, result in modifying one or more routing interfaces of the network routing device to permit communications between the client and the network resource.

* * * * *

(D) Evidence for Claim 16 – Relied Upon

The following item (1) listed below is hereby entered as evidence relied upon by the Examiner as to grounds of rejection for claim 16, to be reviewed on appeal. Also listed for each item is where said evidence was entered into the record by the Examiner.

(1) Copy of US Patent Number 6,549,773 B1 ("Linden et al"). This evidence was entered into the record by the Examiner on page 9 at 11 of the Office Action mailed 08/10/2006.

Copies of all References follows.

//

US006549773B1

(12) **United States Patent**
Linden et al.

(10) **Patent No.: US 6,549,773 B1**
(45) **Date of Patent: Apr. 15, 2003**

(54) **METHOD FOR UTILIZING LOCAL RESOURCES IN A COMMUNICATION SYSTEM**

(75) Inventors: **Mikael Linden**, Tampere (FI); **Teemu Kurki**, Lempäälä (FI)

(73) Assignee: **Nokia Mobile Phones Limited**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/399,579**

(22) Filed: **Sep. 20, 1999**

(30) **Foreign Application Priority Data**

Sep. 21, 1998 (FI) 982031

(51) **Int. Cl.**^{7} **H04Q 7/20**

(52) **U.S. Cl.** **455/426**; 455/412; 455/517; 455/558; 370/328; 370/338; 370/401; 709/230; 709/227; 709/228

(58) **Field of Search** 455/558, 412, 455/517, 426; 370/328, 338, 401; 709/230, 227, 228

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,868,846	A		9/1989	Kemppi	379/144
5,266,782	A		11/1993	Alanara et al.	235/380
5,315,635	A		5/1994	Mukari	379/58
5,353,328	A		10/1994	Jokimies	379/58
5,448,622	A		9/1995	Huttunen	379/58
5,487,084	A		1/1996	Lindholm	375/215
5,600,708	A		2/1997	Meche et al.	379/59
5,956,633	A		9/1999	Janhila	455/410
6,011,976	A	*	1/2000	Michaels et al.	455/466
6,253,326	B1	*	6/2001	Lincke et al.	713/201

FOREIGN PATENT DOCUMENTS

WO	WO 94/30023	12/1994
WO	WO 97/36437	10/1997
WO	WO 98/27767	6/1998

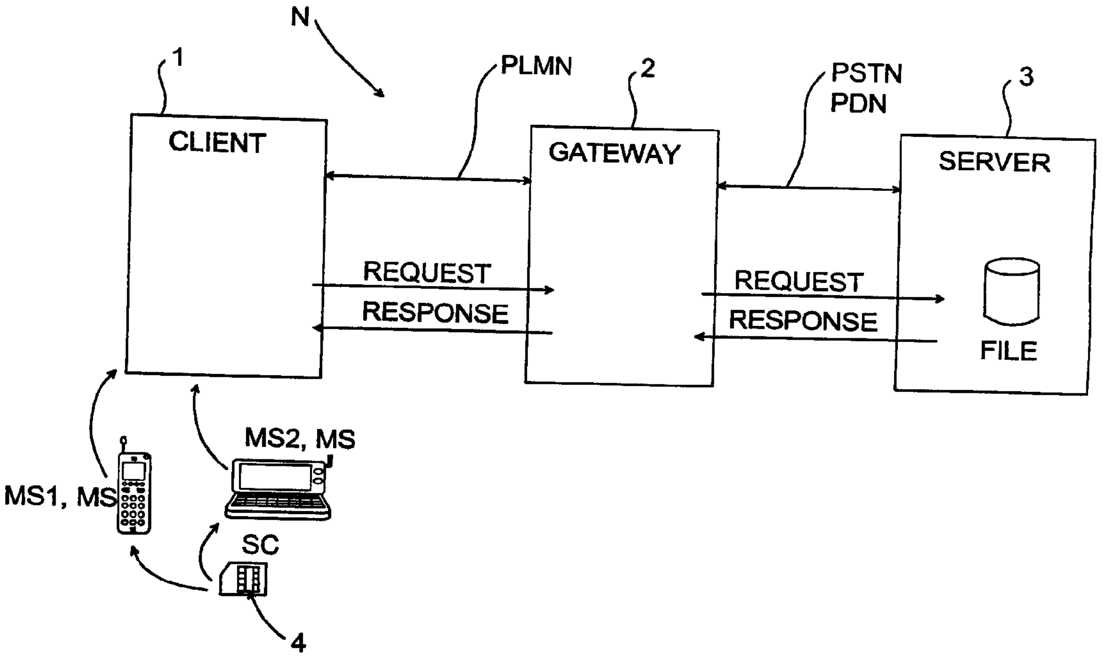
* cited by examiner

Primary Examiner—William Trost
Assistant Examiner—Danh Le
(74) *Attorney, Agent, or Firm*—Perman & Green, LLP

(57) **ABSTRACT**

The invention relates to a method in data transmission between a first mobile station (MS, MS1, MS2), a second mobile station (MS, MS1, MS2), and advantageously also a server (**3**, SERVER), wherein the first mobile station (MS, MS1, MS2) and the second mobile station (MS, MS1, MS2) comprise protocol means (**100–106**) for generating and directing a request (REQUEST) which contains at least address information (URI, URL, URN) for identifying the destination of the request (REQUEST), and wherein at least the first mobile station (MS, MS1, MS2) comprises a first local resource (SC, **4**), such as a smart card, connected to the same. In the information, the address information (URI, URL, URN) of the request (REQUEST) is established to identify said first local resource (SC, **4**), and the request (REQUEST) is generated and directed at least partly with the help of said protocol means (**100–106**).

17 Claims, 7 Drawing Sheets



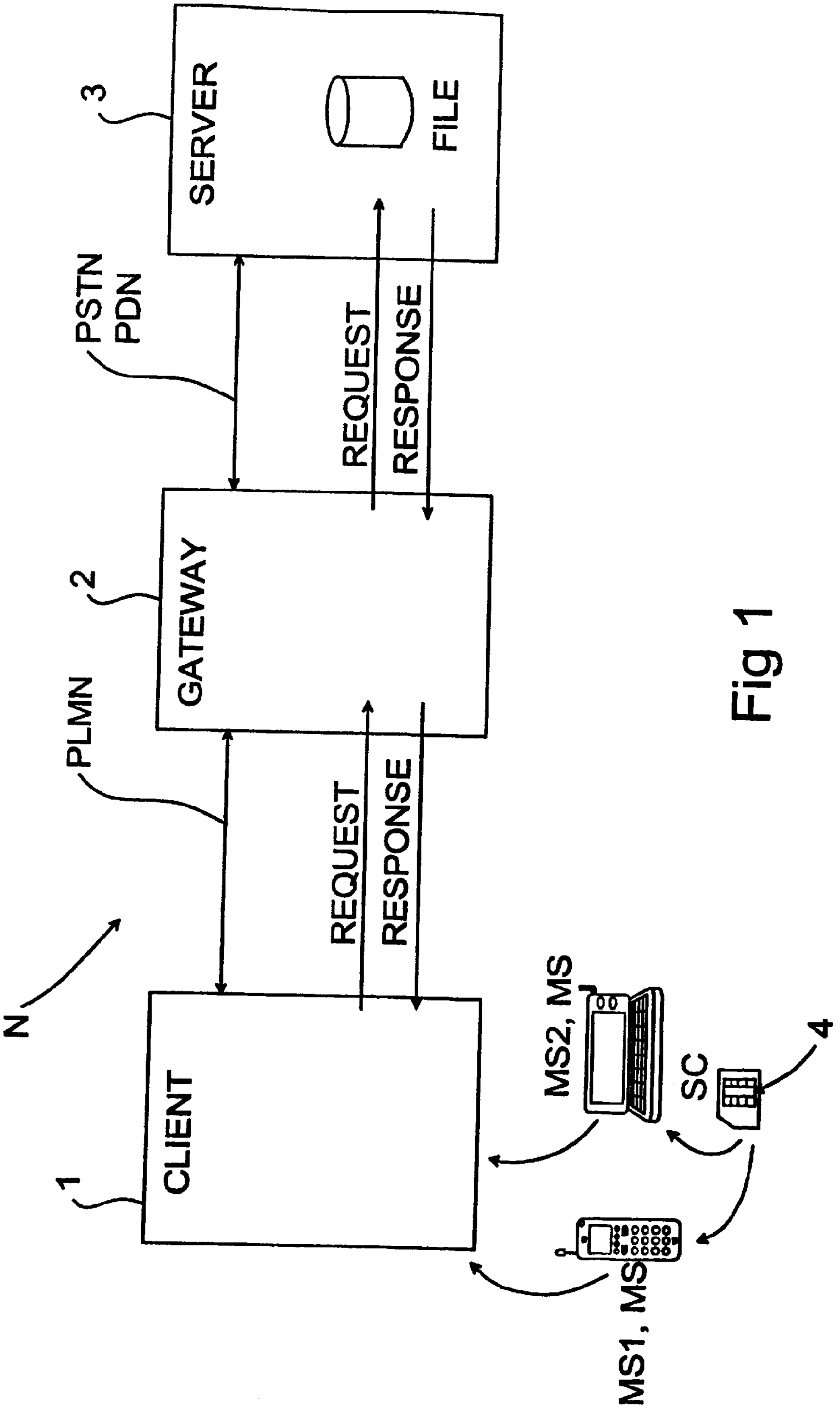

N
CLIENT
1
GATEWAY
2
SERVER
3
FILE
PLMN
PSTN
PDN
REQUEST
RESPONSE
REQUEST
RESPONSE
MS2, MS
SC
4
MS1, MS

Fig 1

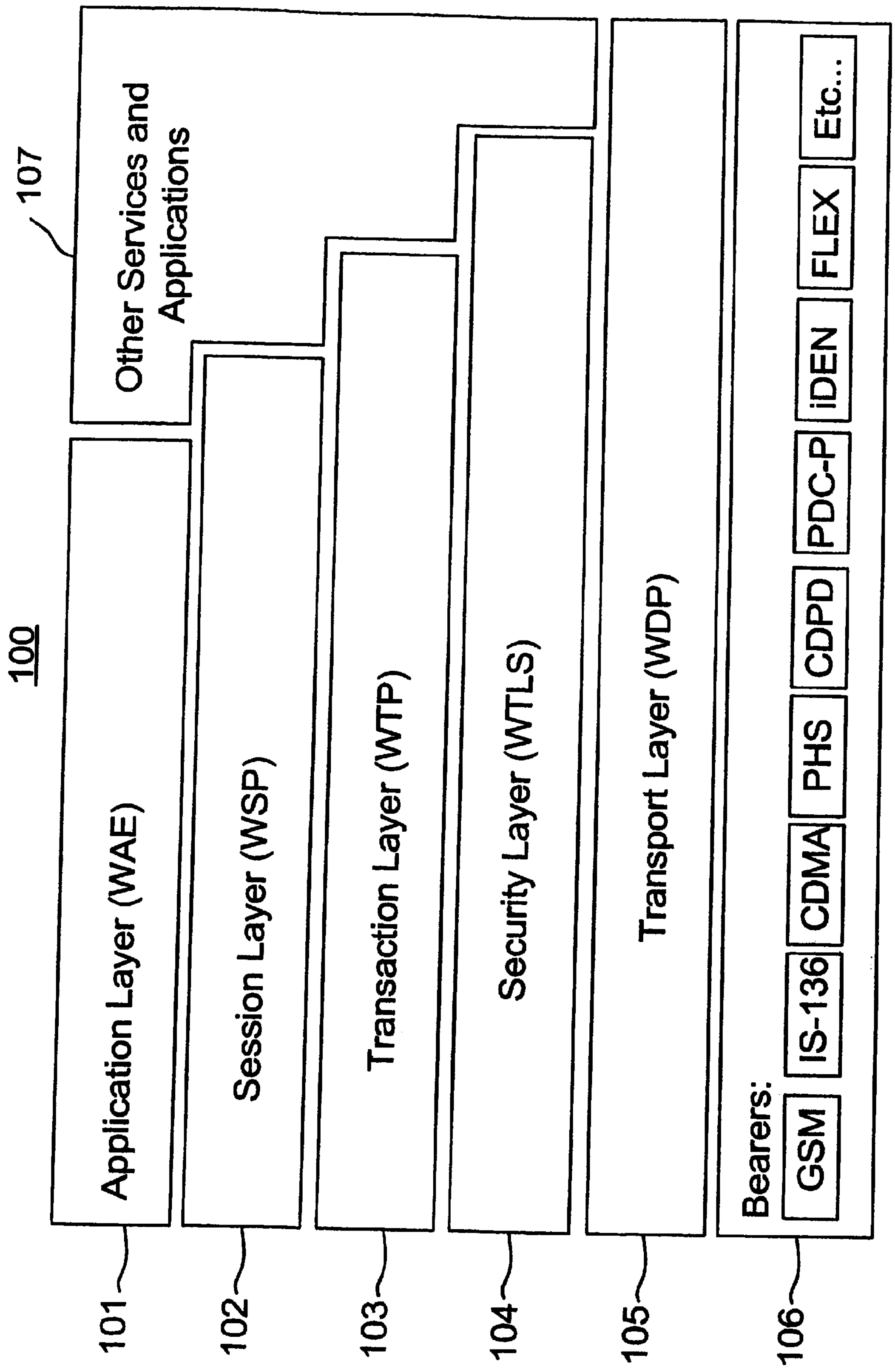

100
107
Other Services and Applications
Application Layer (WAE)
Session Layer (WSP)
Transaction Layer (WTP)
Security Layer (WTLS)
Transport Layer (WDP)
Bearers:
GSM
IS-136
CDMA
PHS
CDPD
PDC-P
iDEN
FLEX
Etc...
101
102
103
104
105
106

Fig 2a

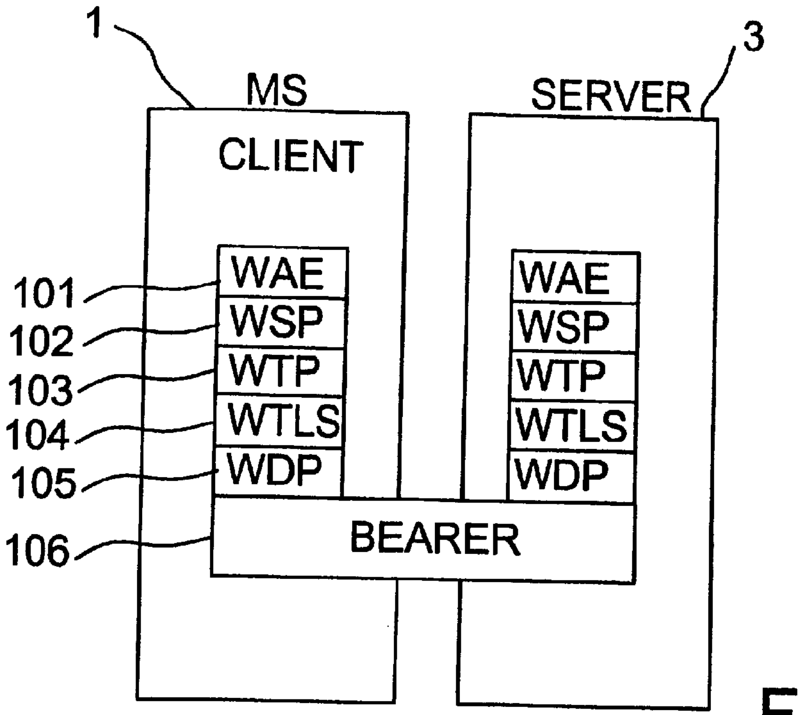

1
MS
CLIENT
3
SERVER
101
102
103
104
105
106
WAE
WSP
WTP
WTLS
WDP
WAE
WSP
WTP
WTLS
WDP
BEARER

Fig 2b

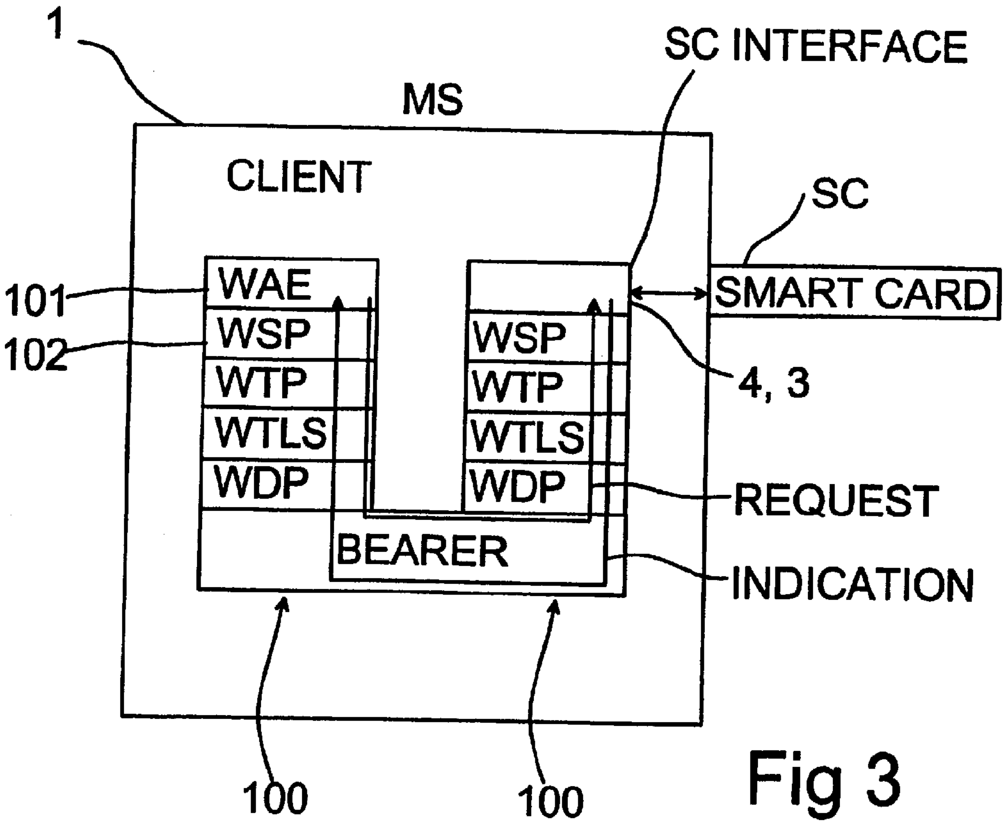

1
MS
CLIENT
SC INTERFACE
SC
SMART CARD
101
102
WAE
WSP
WTP
WTLS
WDP
WSP
WTP
WTLS
WDP
BEARER
4, 3
REQUEST
INDICATION
100
100

Fig 3

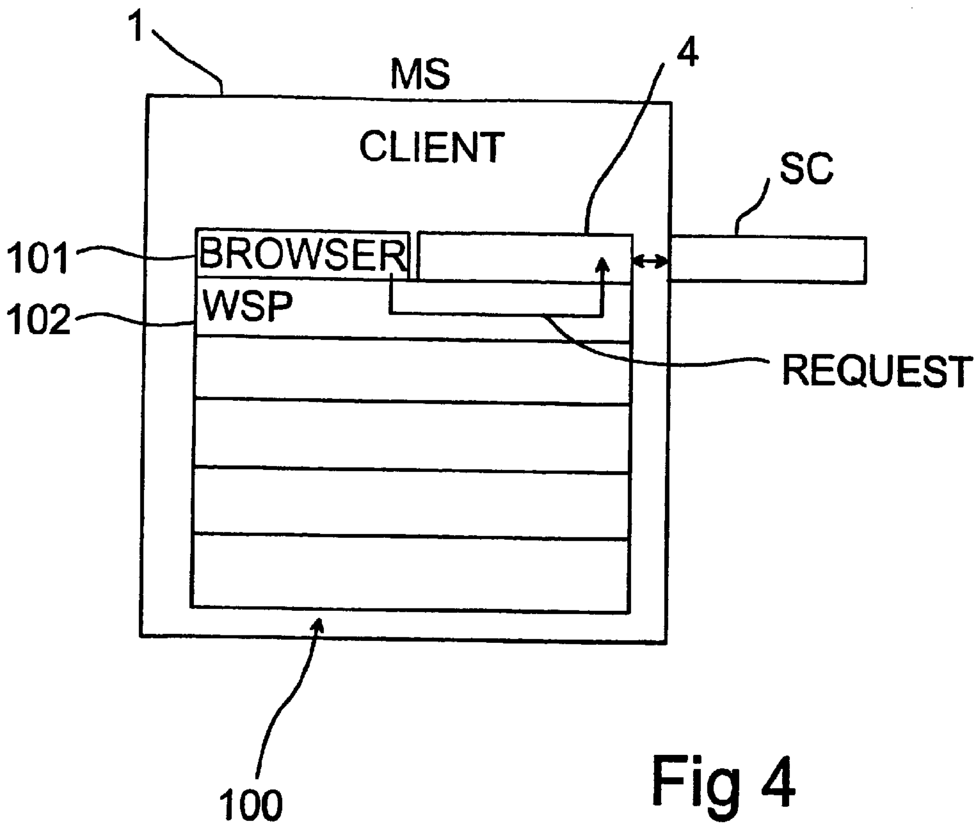

1
MS
4
CLIENT
SC
101
BROWSER
102
WSP
REQUEST
100

Fig 4

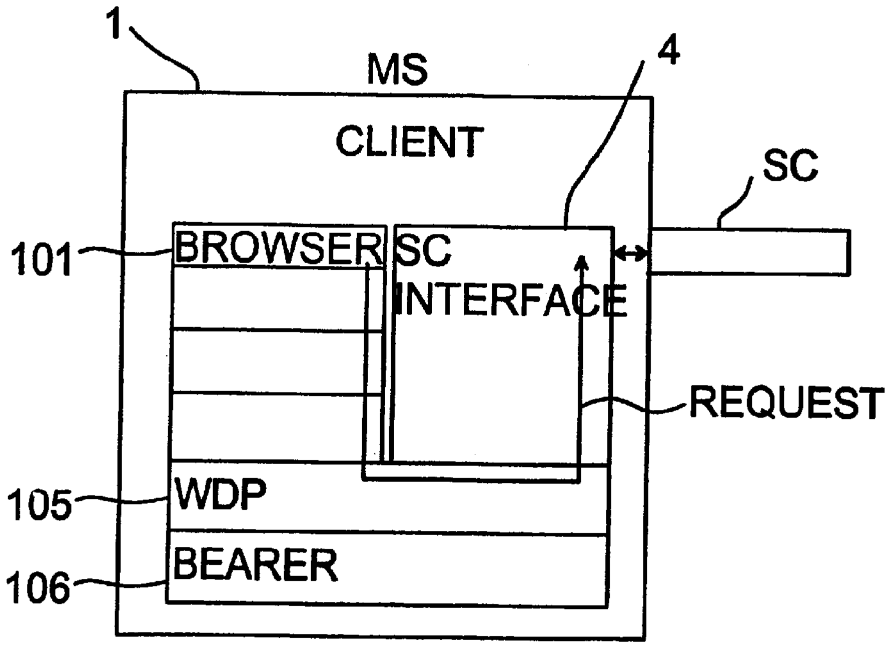

1
MS
4
CLIENT
SC
101
BROWSER
SC INTERFACE
REQUEST
105
WDP
106
BEARER

Fig 5

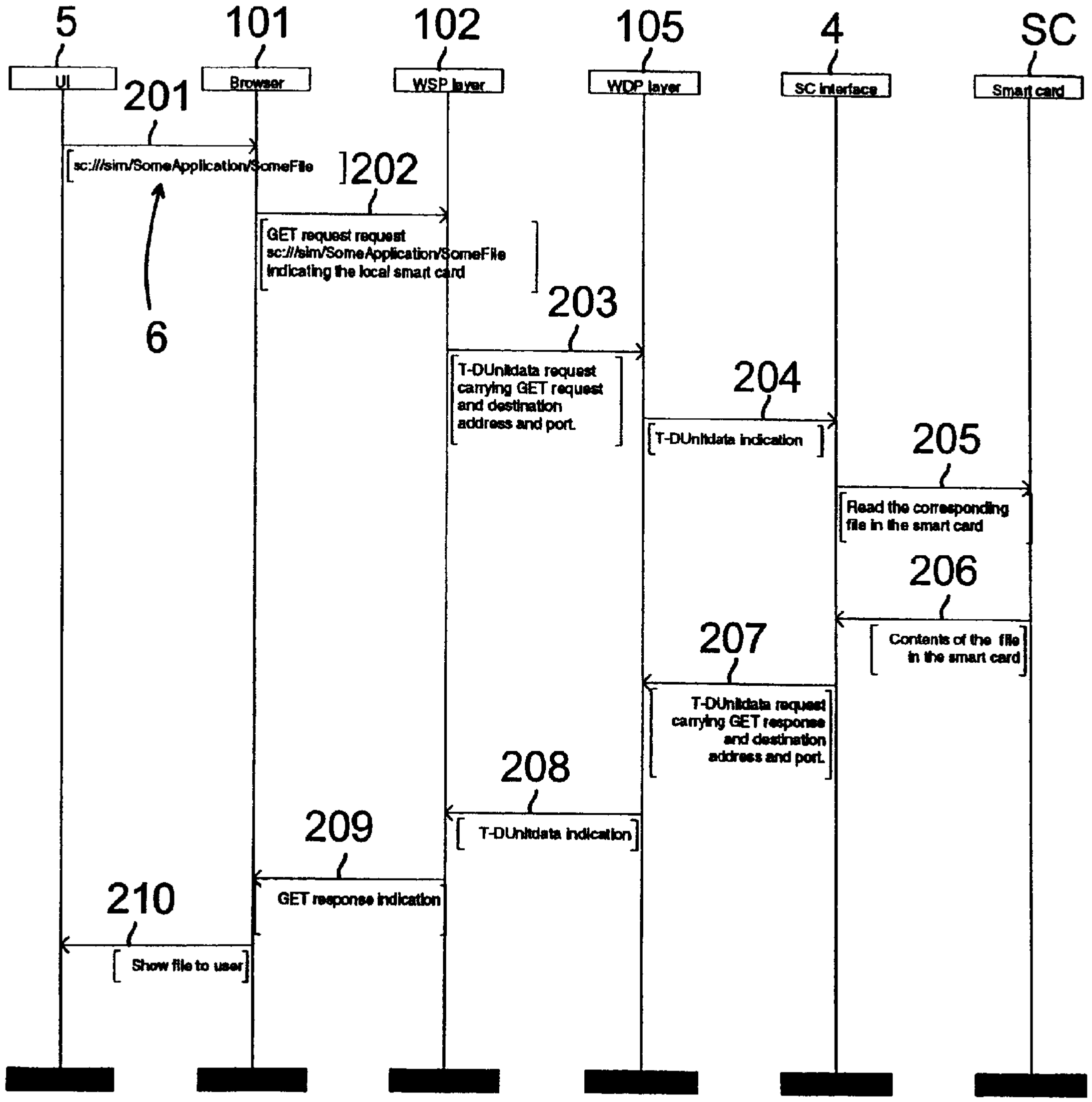

5
UI
101
Browser
102
WSP layer
105
WDP layer
4
SC interface
SC
Smart card
201
sc:///sim/SomeApplication/SomeFile
202
6
GET request request
sc:///sim/SomeApplication/SomeFile
indicating the local smart card
203
T-DUnitdata request
carrying GET request
and destination
address and port
204
T-DUnitdata indication
205
Read the corresponding
file in the smart card
206
Contents of the file
in the smart card
207
T-DUnitdata request
carrying GET response
and destination
address and port
208
T-DUnitdata indication
209
GET response indication
210
Show file to user

Fig 6

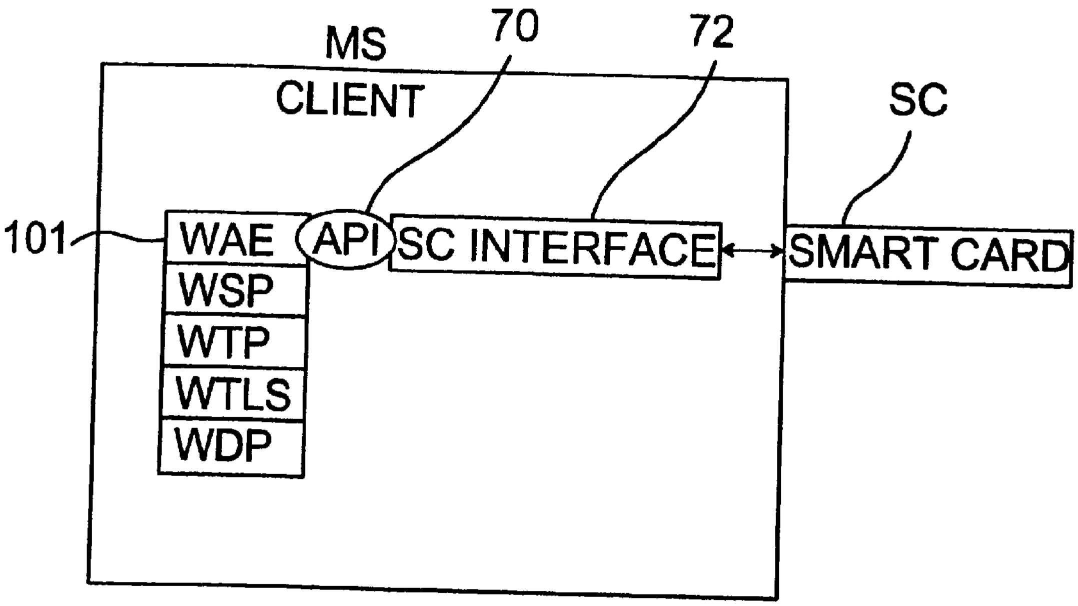

MS
70
72
CLIENT
SC
101
WAE
API
SC INTERFACE
SMART CARD
WSP
WTP
WTLS
WDP

Fig 7a

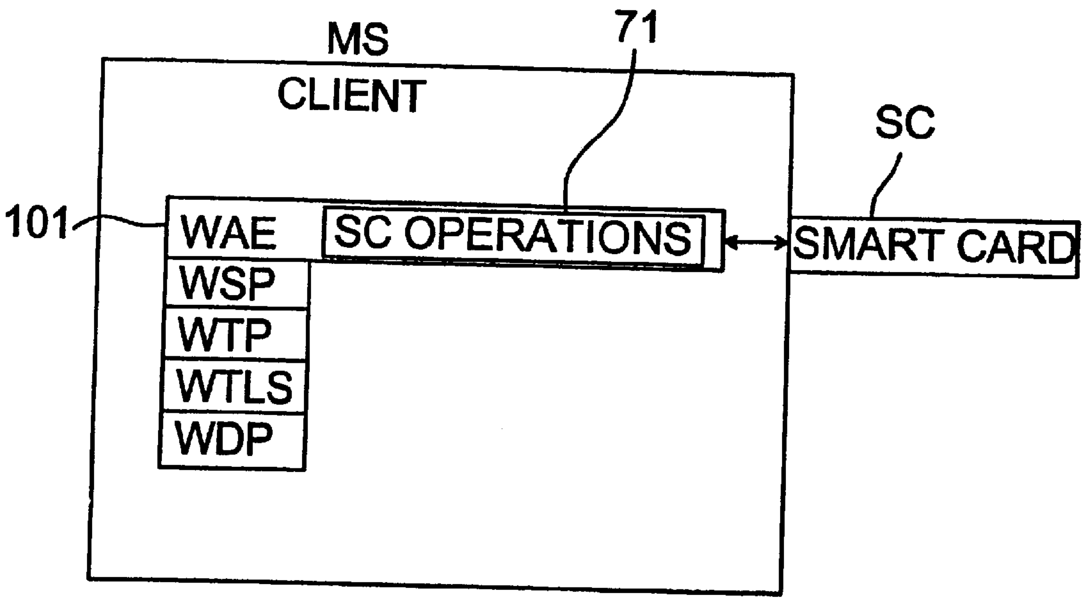

MS
71
CLIENT
SC
101
WAE
SC OPERATIONS
SMART CARD
WSP
WTP
WTLS
WDP

Fig 7b

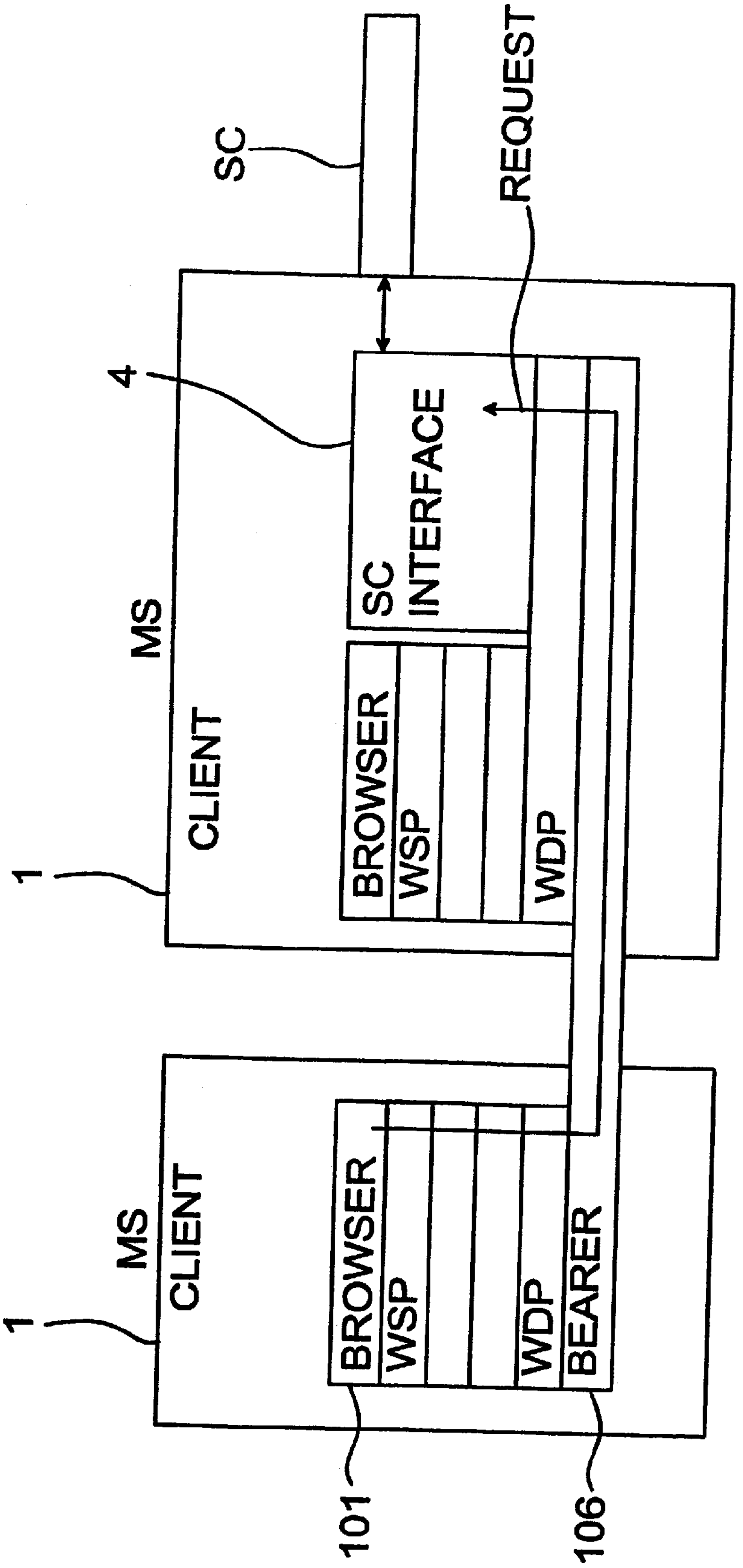

SC
REQUEST
4
SC
INTERFACE
MS
CLIENT
1
BROWSER
WSP
WDP
MS
CLIENT
1
BROWSER
WSP
WDP
BEARER
101
106

Fig 8

METHOD FOR UTILIZING LOCAL RESOURCES IN A COMMUNICATION SYSTEM

The present invention relates to a method in a communication system according to the preamble of claim **1**. The invention also relates to a communication system according to the preamble of claim **11**. Furthermore, the invention relates to a wireless communication device according to the preamble of claim **12**.

There are known wireless communication systems, such as the PLMN (Public Land Mobile Network), which is a communication network based on a cellular system. One example that can be mentioned is the GSM **900** mobile communication network according to the GSM standard (Global System for Mobile Communications). The cells of the communication network are distributed within a wide geographical area, and mobile stations (MS), such as mobile phones, which are connected to the communication network via base stations BS, move from one cell to another. These mobile phones are distinguished from each other by means of a subscriber-specific identification code, wherein communication, such as data transmission or an audio call, is possible between two mobile stations. The identification code is, for instance, an IMSI code (International Mobile Subscriber Identity). The communication network takes care of routing information via base stations and mobile services switching centers (MSC), by utilizing register data which indicate the location of the mobile station in the area of the cells of different base stations. Furthermore, the following wireless communication networks should be mentioned: GSM-1800, GSM-1900, PDC, CDMA, US-TDMA, IS-95, USDC (IS-136), iDEN (ESMR), DataTAC, and Mobitex.

In order to perform data transmission and processes connected with data transmission in communication devices, such as servers and wireless communication devices, which are connected to a communication network, a set of communication rules must be available for defining the allowed messages and the function of the participants of the data transmission at the different stages of the communication. As is well known, one such set of communication rules in data transmission is a protocol used by the devices to communicate with each other. For data transmission especially in wireless communication networks, a wireless application protocol WAP is developed, which will be used as an example in the following specification. One version of the WAP application protocol is specified in the WAP Architecture Version Apr. 30, 1998 publication (Wireless Application Protocol Architecture Specification; Wireless Application Protocol Forum Ltd, 1998), which is published in the Internet, and which includes for example a description on the architecture of the WAP application protocol. By means of the WAP application protocol, it is possible to define a series of protocols on different levels, which can be used to develop new services and devices e.g. for digital mobile communication networks based on a cellular network. For example, the WAP application protocol has already been developed for SMS services (Short Messaging Service), USSD services (Unstructured Supplementary Services Data), CSD services (Circuit Switched Data), and GPRS services (Global Packet Radio System) of the GSM network, and for the services of the IS-136 and PDC network.

The WAP application protocol is designed to describe those standard components that enable data transmission especially between mobile stations (client) and servers of the communication network (origin server). In order to gain access to servers located in the WWW network, the WAP

uses a gateway which also functions as a proxy containing functions for data transmission between a WAP protocol stack and a WWW protocol stack (HTTP, TCP/IP), as well as functions for coding and decoding the content (WML, Wireless Markup Language, or HTML) of the information for data transmission. In the WAP, specified presentation formats are used to define the content of the information and the applications. The content is transferred using standardized data transmission protocols. A so-called browser or a microbrowser is used in the wireless communication device to control a user interface (UI).

The application layer in the architecture of the aforementioned WAP application protocol applies a defined architecture of a wireless application environment WAE. The purpose of the WAE application environment is to provide operators and service providers with an open environment, by means of which it is possible to create a large group of services and applications on top of different wireless communication methods functioning as a platform. The different WAE applications of communication devices follow a procedure used in the Internet World Wide Web (WWW) network, in which different applications and information are presented by means of standardized presentation formats and browsed for example with known WWW browsers. Consequently, in order to use different resources of communication devices, the servers and the information of the Internet network are labelled with a URI address (Uniform Resource Identifier) which is independent of the location, and the presentation format of the information is supported by the browser used, and is, for example, HyperText Markup Language (HTML) or JavaScript. On the other hand, the WAE application environment especially takes into account the requirements of the wireless communication devices and wireless communication networks. At present, according to prior art, the WAE applications (user agents), such as browsers, only support the WSP/B protocol. For example browsers communicate with a gateway server via a WSP layer (Wireless Session Protocol) of the WAP protocol stack, which will be described later. The gateway, in turn, provides functions for converting the data transmission protocol so that access to the resources of a WWW server using the HTTP protocol would be possible. According to prior art, the WAP protocol stack is described in more detail in the aforementioned publication, and the WAE application environment is described in more detail e.g. in the WAP WAE Version Apr. 30, 1998 publication, published in the Internet (Wireless Application Protocol Wireless Application Environment Overview; Wireless Application Protocol Forum Ltd, 1998).

The URI addresses are used to locate resources by providing the location of the resource with an abstract identification. When the resourse is located, the system can subject the resource to different procedures which depend on the application and on the purpose of pursuing access to the resource. As is well known, several different data transmission protocols are used in this context, of which for example the HTTP (HyperText Transport Protocol), FTP (File Transfer Protocol), MAILTO (Electronic Mail Address), and GOPHER (The Gopher Protocol) can be mentioned.

As is well known, the URL address used by the HTTP is utilized to indicate resources which are available in the Internet network, for example in its servers, by using an HTTP data transmission protocol, and it has the format:

http://<host>:<port>/<path>,

in which the data transmission protocol used can be deduced from the "http" part, "<host>" represents the domain name or the IP address (Internet Protocol) of the server in the communication network, "<port>" represents the number of the port, and it can also be left out, because data transmission protocols use a default port. Furthermore, "<path>" describes the resource in question in more detail and functions as a selector in the HTTP. The prefix "//" illustrates that the address follows the data transmission protocols used in the Internet network. A more precise indication of the resources by means of the "<path>" part varies in different data transmission protocols, and in addition to that, it is possible to provide a "<user>" part between the "//" and "<host>" parts to indicate the user, as in the FTP, and a "<password>" part to indicate a password. The resource can also be identified by means of a URN name (Uniform Resource Naming), wherein it is possible to use only a name instead of a URL address, and the URN name is modified to a URL address when necessary. The URI address and the protocol (access algorithm) to be used, form a URL address (Uniform Resource Locator) to identify the resource.

As is well known, resources refer to a area (such as a directory or a file), a program (such as an application), or a peripheral device (such as a printer) allocated in the server of the data transmission network for collective use. In the prior art, however, problems often occur which relate especially to the use of these local resources. These local resources can include for example content and files, as well as applications contained in the same communication device, and peripheral and auxiliary devices connected to the same. Local resources can also be located in a peripheral device, such as a SIM card (Subscriber Identity Module) or a smart card, connected to a wireless terminal, such as a mobile phone. At present, for example the WAP application protocol contains no specification for methods for utilizing these local resources coupled to a wireless communication device.

In connection with WWW communication networks, there are such known agents as for example the WWW browsers Netscape Navigator and Microsoft Internet Explorer. A function known from the Navigator browser is the viewing of local files which are stored in the local hard disc of a data transmission device, typically a computer. The browser opens a file stored on the hard disc, reads its content and presents it in an intelligible format to the user. This browsing of files is implemented as an extension to the browser. Consequently, all necessary command primitives and references to the services (i.e. system functions) provided by the operating system software controlling the actual operation of the device, are built in the program code of the browser in order to implement the browsing of files. This can be implemented for example by means of a so-called standardized API interface, i.e. an application programming interface.

In the WAE application environment, the above-described facts would entail for example that an API interface was arranged for each WAE application of a wireless communication device for coupling to a smart card. Correspondingly, the interface of the smart card would be responsible for transmitting low-level commands (command APDU) to the smart card. The responses given by the card are then transmitted back to the application via interfaces. Known methods to attend to the coupling are the Open Card Framwork specification and the PC/SC specification. However, the specifications differ from each other, and thus each WAE application should take the differences and various API interfaces into account.

Another possibility is to arrange the WAE application in such a way that it would be in direct communication with a resource, such as a smart card, which is physically coupled to the mobile station. This would mean that the WAE application would be responsible for transmitting command primitives (command APDU), the functions being programmed in the WAE application. This would create a significant problem, because it would be necessary to provide each WAE application with the necessary command primitives for each different local resource. This would also have the result that for new resources, the WAE applications would have to be supplemented with the necessary command primitives so that these resources could be implemented. That would create a significant need to update the WAE applications, a limited possibility to use the new resources, and a barrier to the implementation and development of new resources.

The purpose of the present invention is to provide a method that enables applications utilizing the used application protocol to gain efficiently access to local resources in a straightforward and efficient manner. A particular purpose of the invention is to introduce a new method to enable applications applying the WAP application protocol to gain access to local resources, which include for example a SIM card of a mobile station, or an attachable smart card.

The central principle of the invention is to utilize the protocol stack used in order to utilize local resources. Another central principle of the invention is to apply an indication corresponding e.g. to the URL address when referring to local resources.

The method in the communication system according to the invention is characterized in what will be presented in the characterizing part of the appended claim 1. The communication system according to the invention is characterized in what will be presented in the characterizing part of the appended claim 11. The wireless communication device according to the invention is characterized in what will be presented in the characterizing part of the appended claim 12.

A considerable advantage of the invention is, that it is possible to avoid including a standardized API interface or necessary command primitives in the application, and nevertheless, local resources can be utilized flexibly. Thus, for example the complexity of WAP applications is reduced considerably, and the implementation of applications in the mobile station is facilitated as well. The changes required to implement the invention in WAE applications are small, because the mechanisms already available in the WAE application (GET and POST methods, etc) are utilized. In the mobile station, the changes required by the invention can be advantageously implemented with changes in the application software which controls the functions of the mobile station.

The principle of the WAP application protocol is that a mobile station functioning as a client contacts a server which is located in the communication network. The advantage of the invention is that this principle can still be applied, wherein to gain access to the local resources, the application makes a request to the lower protocol layer and also receives a response which the server, in this case a local resource, has generated and transmitted to the client. A further advantage of the invention is that the principle can also easily be expanded for example in such a way that the local resource can receive requests from another mobile station or server too, not only from the one containing the smart card. In that case, the interface of the smart card functions like a server.

The local resource can also transmit requests to other mobile stations and servers, and, furthermore, also to the mobile station to which it is attached. The advantage therein is that the smart card, such as a SIM card, can transmit a request to the server, for example to update applications functioning in a SIM card from the server of the operator (download).

The principle of the invention can also be implemented in a situation where the smart card is communicating with another smart card. An example that can be mentioned is the transfer and relay of payments from one smart card to another. Thus, one of the smart cards can be connected to a server functioning in the network, and a protocol stack according to the invention is implemented in the server. By means of the "<host>" part of the URL address, it is possible to indicate said server. The user attaches the smart card into the mobile station, after which an application is activated for setting up a connection to the server, the application being loaded from the server and implemented for example by means of a WMLScript command language.

A particular advantage of the invention e.g. in connection with the WAP application protocol is that it is possible to efficiently utilize functions connected with the HTTP data transmission protocol of the WSP/B protocol already known as such. These include, for example, GET, PUT, and POST requests. Consequently, the header fields of the HTTP protocol can also be utilized in the data transmission, as well as the headers of the HTTP protocol for authentication. Correspondingly, it is possible to utilize efficiently the methods of the WWW communication network for authorization or data transmission.

A further advantage of the invention is that it can be used to conceal the structure and implementation of the local resource from the user of said resource. When several implementations of the local resource, for example a memory module, a smart card or both of them, are coupled to the mobile phone, their lower level interfaces typically are very different. By means of the invention it is possible to use them in the same way, because both resources appear to the user as servers with which communication is effected by means of the WSP/B protocol and via the interface (Local Resource Interface). Thus, the user does not have to understand the real structure of the resource, and the low level interface through which the communication takes place.

In the following, the present invention will be described in more detail with reference to the appended figures, in which

FIG. **1** is a reduced diagram showing a communication system applying the invention,

FIG. **2***a* shows a protocol stack of an application protocol applying the invention,

FIG. **2***b* is a reduced diagram showing the application and logical structure of a protocol stack in the application protocol presented in FIG. **2***a* in implementing the client/server hierarchy,

FIG. **3** is a reduced diagram showing the application and logical structure of the client/server hierarchy presented in FIG. **2***b* in the same wireless communication device,

FIG. **4** is a diagram showing an implementation according to a first preferred embodiment of the invention,

FIG. **5** is a diagram showing an implementation according to a second preferred embodiment of the invention, and

FIG. **6** is a sequential diagram showing how a request and a response are generated and directed according to the invention in the embodiment according to FIG. **5**,

FIGS. **7***a*–**7***b* illustrate alternative implementations for utilizing a local resource in connection of the protocol stack presented in FIG. **2***a*, and

FIG. **8** is a diagram showing an implementation according to a third preferred embodiment of the invention.

FIG. **1** is a reduced diagram showing a communication system known as such in which it is possible to apply the invention. Communication devices, i.e. wireless communication devices **MS1** and **MS2**, advantageously mobile stations (MS) function as clients **1** and are connected to a gateway **2**, which is advantageously a server and which adapts the different data transmission protocols used to each other. The clients **1** utilize advantageously a public land mobile network (PLMN), such as the GSM network and the GSM GPRS network, in order to implement wireless data transmission. The base station subsystem (BSS) of the mobile communication network (PLMN) is known as such and comprises base transceiver stations (BTS) and base station controllers (BSC). The mobile station **MS1**, **MS2** communicates with a base transceiver station via a radio channel, and the base transceiver station communicates further with a base station controller. The base station controller, in turn, communicates with a mobile services switching center (MSC). Mobile services switching centers can, in turn, communicate with each other and with servers in a public switched telephone network (PSTN). The base station controller can also communicate with a public packet data network (PDN). The aforementioned origin or content server **3** can be located either in the PSTN network or in the PDN network, wherein the gateway server **2** uses these networks in data transmission. The server **2** can communicate either with the base station subsystem or with the mobile services switching center in the PLMN network. Thus, the server **2** can be located either in the PLMN network itself or in the PSTN network, and it is obvious that the content server **3** and the gateway server **2** can be physically located in the same communication device. Furthermore, it is obvious that to implement the gateway **2**, several separate servers can be used.

The invention can also be applied in a communication system in which the communication between different data transmission devices, such as servers **2** and clients **1**, takes place by means of short distance IR data transmission (infrared), LPRF data transmission (Low Power Radio Frequency), SDRF data transmission (Short Distance Radio Frequency), or inductive data transmission, wherein data transmission distances in the range of a single communication network are typically shorter than in the mobile communication network.

Smart cards are typically small cards manufactured in the size of a credit card, which, laminated in plastic, contains a micro controller as well as electronic circuits and memory circuits required for the function of the micro controller. Furthermore, the surface of the card typically contains electrical contacts, via which it is possible to transmit operating voltages to the card and to transfer control and data signals between the card and a reading/writing device for the card. There are also known methods in which signals and operating voltages of the card are transmitted wirelessly, for example as high frequency electromagnetic signals, between the card and the reading/writing device for the card. Smart cards are used for example as charge cards in various applications, for example in public telephones, as change cards, as a means of payment in public transport, etc. Asmart card used in mobile phones is a so-called SIM card, which in modern mobile phones is typically a small-sized mini-card inserted in the telephone. The function of the card is, for instance, to store subscriber data and identifications (PIN, Personal Identification Number), and thus the card determines the subscriber number of the telephone. The SIM card can also contain a stored list of telephone numbers or a group of short messages, as well as various data and set values related to the communication network used. This data is utilized increasingly by different applications at the same time when the quantity of data to be stored in the smart card is increased and diversified.

In the following specification, the smart card is used as an example of a local resource in connection with which it is possible to apply the invention. It is, of course, obvious that within the scope of the claims, the invention can also be applied to utilize another local resource, in order to avoid the above described problems and to gain advantages.

FIG. **1** also presents a smart card SC, known as such, which is a mini-card, i.e. a SIM card to be inserted in a mobile station. The smart card SC comprises means known as such which communicate with each other and which are intended to control the functions of the smart card SC and to implement data transmission (not shown in the figure). These means comprise, for example, a control unit (CPU) for controlling the function of the smart card on the basis of a program code stored in a program memory (ROM), and in a data memory (EEPROM) it is possible to store various user-specific data. During the function of the smart card SC, the random access memory (RAM) can be used as a temporary storage location for data. A bus adapter (DATA-I/O) on the smart card SC adapts the communication device functioning as a device for reading the smart card SC, for example a mobile station **MS1**, **MS2**, to connection lines, and to a control and data line. Physically, the smart card SC is coupled to the contacts available in the mobile station typically via its electrical contacts **4**. The features and the purpose of the smart card SC can be set by storing an application software corresponding to the purpose of use, in the program memory of the card SC advantageously at the manufacturing stage of the card. However, new technologies that make it possible to download applications to the smart card are being developed. Also, the protocols related to the interface of the smart card SC in connection with data transmission are taken care of by the application software used.

Further referring to FIG. **1**, the mobile station MS1 and MS2, for example a mobile phone, also comprises (not shown in the figure) means, known as such, for establishing a data transmission connection to the mobile communication network, means for reading the data of the smart card SC, such as a SIM card, and for storing the data on the SIM card, a control unit (CU) for controlling the different functions of the mobile station, which control unit advantageously comprises a micro controller unit (MCU) and a control logic circuit, such as an ASIC circuit (Application Specific Integrated Circuit). The functions include for example controlling the display and reading the keypad. The control unit also contains a memory attached to it, such as a read-only memory (ROM) and a random access memory (RAM). The function of the mobile station is controlled by means of an application software, which is responsible e.g. for implementing the protocols used in data transmission. The function of the mobile station is prior art known by anyone skilled in the art, and thus it is not necessary to describe it in more detail in this context. Known devices include the Nokia 8110, 6110 and 3110 mobile phones. As is well known, there are also devices available which contain two different user interfaces, for example the user interfaces of a mobile phone and a PDA device (Personal Digital Assistant). One such known device is the Nokia 9000 Communicator.

The aforementioned WAP application protocol is used in the following specification as an example of data transmission protocols to clarify the method according to the invention which is the object of this specification. In the following specification, said WAP clients and WAP servers refer advantageously to the clients and servers applying the WAP application protocol in the communication network. It is, of course, obvious that within the scope of the claims, it is also possible to apply the invention in connection with another application protocol, wherein the WAP indication mentioned in this specification will be used to refer to the use of this application protocol.

With reference to FIG. **2***a*, the protocol stack **100** (WAP Protocol Stack) in an OSI layer model of an advantageous WAP compatible system contains the following layers listed downwards from the top layer:

- an application layer **101** for the wireless application environment WAE, which also comprises functions for a browser, which functions comprise a wireless markup language (WML), a WMLScript command language, and WTA services and WTA interfaces for telephone functions and programming interfaces, as well as content formats necessary for presenting information,
- a wireless session protocol (WSP) of a session layer **102**,
- a wireless transaction protocol (WTP) of a transaction layer **103**,
- a wireless transport layer security protocol (WTLS) of a security layer **104**, intended to be used in connection with a WAP transport protocol,
- a transport layer **105** for data packets of a wireless datagram protocol (WDP), which data packets also contain address information of the destination and other necessary information in addition to the actual data,
- different bearer services **106**, which include for example transmission of short messages, circuit-switched data transmission and packet-switched data transmission.

To describe the different functions, the layer model advantageously refers to an ISO/OSI layer model known as such. The upper layers (WAE, WSP, WTP, WTLS) of the WAP architecture are independent of the data transmission network used, but the WDP layer **105** has to be applied according to the data transmission method used at a time, for example on the basis of the GSM network or special requirements. The WAP protocol stack also allows other services and applications **107** to utilize the WAP stack by means of specified interfaces.

FIGS. **7***a* and **7***b* illustrate the above-described alternative methods for utilizing a local resource. However, these methods entail problems which the invention aims to eliminate. FIG. **7***a* presents the use of a fixed API interface **70** to utilize a smart card SC, for example to read the content of a file contained in the smart card. In this context, an advantageously standardized interface **72** of the smart card SC is utilized. FIG. **7***b* presents another alternative, in which the necessary functions **71**, advantageously references to the system functions of the smart card SC, are included in a WAE application **101** to utilize the smart card SC directly by means of the application **101**. By means of the invention, however, the aim is to utilize the functions and definitions of the protocol stack **100** presented in FIG. **2***a*.

The purpose of the data transmission network and system is to provide the clients and servers located in the network with a communication channel, wherein the protocol stack **100** is utilized at the same time according to FIG. **2***b* in a way known as such. FIG. **2***b* presents the described logical structure of the client/server hierarchy, which comprises a client **1** located in a mobile station MS and a server **3** located in a communication network. WAE applications **101** (i.e. clients **1**) make a request to the WSP layer **102** by means of available command primitives. The layer **102** in question transmits the request further to the server **3**.

FIG. **3** presents the application of the invention, wherein the server **3** (i.e. the interface **4** of the smart card) is located in the same device MS with the client. Logically, the server **3** seems to be located in the communication network, wherein the request made by the client **1** is conducted according to the invention as if the server in question was a conventional server. According to the invention, the protocol layers are responsible for transmitting the request to the smart card SC interface **4**. WAE applications **101** use the service primitives provided by the WSP layer **102** to transmit requests to the server **3**. The server **3** processes the request and transmits a response to the client **1** (i.e. to the application **101**). The primitive types of these service primitives can be divided for example into following types known as such:

the request conducted by an upper layer to obtain services from a lower layer,

the indication by means of which the layer providing services notifies an upper layer e.g. of a request made by a client or an activity initiated by the protocol,

the response, by means of which an upper layer notifies a lower level that it has received an indication,

the confirmation, by means of which the layer providing services confirms the maker of the request that the function is completed successfully.

The primitive types describe logical data transmission between adjacent layers in an abstract manner, and, unlike the API interface, they are not used to describe the actual implementation in detail.

According to FIG. **3**, the use of two protocol stacks **100** in the same device MS is not necessary when implementing the invention, but it is possible to use one protocol stack **100** according to FIG. **4**. According to the first preferred embodiment of the invention, it is possible to notice in the WSP layer **102** that the URL address, i.e. the network address, used by the application **101**, advantageously a browser, refers to the smart card SC, wherein requests can be addressed directly to the smart card interface **4**. With reference to FIG. **5**, according to the second preferred embodiment of the invention, the interface **4** of the smart card can be logically located above the WDP layer **105**, wherein the port numbers included in the request can be used to separate the smart card SC and the applications **101** of the communication device from each other. Thus, a separate port number i.e. a port is allocated for the interface **4** of the smart card.

The interface **4** of the smart card can also be logically located above the WTLS layer **104**, wherein it is possible to utilize the functions of the security layer **104** in request transmission. Correspondingly, the interface **4** of the smart card can also be located above the WTP layer **103**, especially in the case of connection-oriented data transmission. In both aforementioned cases, a separate port number is allocated for the interface **4** of the smart card.

Further referring to FIG. **5**, according to a preferred embodiment of the invention, the WAE application **101** transmits a request to a local resource, advantageously to the smart card SC, by using the service primitives of the WSP layer **102**. Correspondingly, the interface **4** of the smart card conducts the request to the smart card SC. The smart card SC gives a response to the interface **4** of the smart card, which generates a necessary response and transmits it to the WAE application **101**. In more detail, the smart card SC is defined as a new server **3**, which responds to requests related to the smart card SC. Thus, with respect to the WAE application **101**, the smart card SC is like any server **3** located in the communication network, wherein logical transaction corresponds to the normal transaction of the protocol stack **100** according to FIG. **3**. Requests are not, however, always directed to the communication network, but lower layers direct the requests to the interface **4** of the smart card, which can be located physically in the same device (MS) with the WAE application.

The WSP layer provides means for exchanging the information content between the applications of the client **1** and of the server **3**. The services and protocols defined (WSP/B, Wireless Session Protocol/Browsing) are adapted especially for browser type applications. The WSP/B contains protocols both for a transport service of packets (datagram) in connectionless data transmission and for a transaction service of the session service in connection-oriented data transmission. The WSP/B follows closely the definitions of the HTTP data transmission protocol and supports applications which are HTTP/1.1 compatible. For example, methods such as GET, PUT and POST used by the HTTP can be used to retrieve (GET) or to transmit (PUT, POST) information. The header fields of the HTTP protocol can be utilized for giving information related to the content type of the message. It is also possible to use the header of the HTTP protocol for authentication. Correspondingly, the methods of the WWW communication network for authorization and data transmission can be utilized efficiently.

The invention is implemented in the WSP layer in such a way that the data transmission protocol used is deduced from the URL address, for example in this context "sc://", referring to the smart card, and the server used, described by the "<host>" part of the URL address. The requests whose address information begins with a reference "sc:///", are, according to the invention, directed to the interface of the smart card. According to the first preferred embodiment of the invention, and with reference to FIG. **4**, this is effected via the WSP layer **102**, and according to the second preferred embodiment of the invention and with reference to FIG. **5**, this is effected via the WDP layer **105**.

According to a preferred embodiment of the invention, the URL address to be used is formulated in the following way when the local resource is a smart card:

sc://<host>:<port>/<url-path>,

in which the "sc://" part refers to the smart card and the "<host>" part to the device to which the smart card is connected. When the "<host>" part is omitted, it can be assumed that the address refers to a smart card connected physically to the same mobile station. The "<port>" part can also be omitted, if a default port is used. Since a mobile station can simultaneously contain various smart cards, which, in turn, contain several different files which one wishes to browse or retrieve information from, these are separated from each other by means of the "<url-path>" part, for example in the following way:

sc:///sim/_7F2A/_6FO5.

In the following, an implementation of the invention according to FIG. **5** will be described, in which the smart card interface **4** is located above the WDP layer **105**. The WDP layer **105** is located above bearer services **106** which are supported by different communication networks (GSM, GSM GPRS, etc). In this context, for example the communication system according to FIG. **1** will be utilized in a way known as such. The WDP layer **105** provides services for upper layers **101–104**, which can thus be in transparent communication via the available bearer services **106**. By means of the used port number, an upper layer entity is identified, which can be a WTP transport protocol **103**, a WSP session protocol **102** or an application **101**.

FIG. **6** presents a command sequence for using a local resource, especially a smart card SC, by means of a WAP application **101** (browser) particularly in connectionless data transmission. In the example, a request is made by means of a GET method to a SIM card **4** coupled to a mobile station MS. By means of a user interface (UI) **5**, the user enters (stage **201**) in a WAE application **101**, which is e.g. a browser, a determined URL address **6**, in which the "sc:///" part refers to the smart card SC, i.e. SIM card **4**, coupled to the mobile station MS in question. The address entered by the user has, for example, the format "sc:///sim/SomeApplication/SomeFile". At the next stage **202**, the browser **101** calls the service primitives of connectionless data transmission in the WSP layer **102**, in order to transmit a GET request to the smart card interface **4**. Here, the WSP layer **102** uses (stage **203**) a corresponding command primitive of the WDP layer **105**, in order to transmit a WSP/B request to the smart card interface **4**. A destination address used in the request refers to the mobile station MS itself, and a destination port number refers to a default port number allocated for the smart card interface **4**. The request used in FIG. **6** (stage **203**) is an service primitive T-DUnitdata, known as such, which is used to transmit data in a datagram, and by means of which it is possible to transmit parameters describing the destination address, the destination port, i.e. the address of the application **101** connected to the destination address, the source address, the source port, and user data transmitted by the WDP protocol **105**. In the request, all this information has to be given, but in an indication, the destination address, the destination port and the user data are advantageously sufficient. The destination address can also be an individualizing MSISDN number, an IP address, an X0.25 address, or another identification, known as such. The parameters are transmitted in a way known as such in packet transmission, wherein the packet contains for example header information and data, which information is coded in the packet into bit sequences of fixed size, typically octets, which are transmitted by means of the data transmission method used.

With reference to FIG. **6**, at the next stage, the WDP layer **105** detects that the destination address belongs to a local mobile station MS, i.e. to the wireless communication device MS, to which the smart card SC is coupled as well, wherein the indication is transmitted (stage **204**) to the appropriate port in the smart card interface **4**. The smart card interface **4** processes the indication and processes the WSP/B request by making the necessary requests to the smart card SC (stage **205**). After this, the smart card SC gives the content of the desired file to the smart card interface **4** (stage **206**), after which the interface **4** encapsulates the content in the WSP/B response and transmits (stage **207**) it as a request to the WDP layer **105**. At the next stage **208**, it is detected in the WSP layer **102** that the destination address and the destination port of the request belong to the browser **101** of the local mobile station MS, and the indication is transmitted (stage **208**) to the WSP layer. After this, the WSP layer **102** transmits (stage **209**) a WSP/B response to the browser **101**, and the browser **101** presents the content of the file to the user (stage **210**), advantageously by using the user interface **5**. The aforementioned WSP/B requests and responses can contain necessary header information, related, for example, to the content type of the message and to the authentication and authorization of the user. Furthermore, they can include data relating to the compression method used and data for parity checking.

In the above-described example, the user activates the use of local resources by giving a URL address by means of the user interface of the mobile station, which user interface is implemented in a way known as such by utilizing the display and keypad of the mobile station. In a preferred embodiment of the invention, the application itself can effect the activation, typically in order to retrieve information stored in the memory of the SIM card. Furthermore, it is obvious that according to the invention, also a server coupled to the communication system can request information from the smart card coupled to the mobile station. FIG. **8** illustrates the invention further in its third preferred embodiment, wherein a local resource, for example a smart card SC, receives requests also from clients **1** (i.e. from browsers **101**) other than those located in the same mobile station MS with the smart card SC. In this case, the interface **4** of the smart card operates like a server located in a network. In these different cases, the method already presented in FIG. **6** will be applied, which method can be implemented by anyone skilled in the art on the basis of the presented example. In this context, it has to be noted that the method also utilizes bearer services **106** according to FIG. **8** to transmit a request between mobile stations. The communication network available is also utilized therein.

The present invention is not restricted solely to the above described examples, but it can be modified within the scope of the appended claims. For example, the presented protocol stack can be implemented in a wireless communication device in which data transmission or the communication system is based on the previously mentioned IR, LPRF, SDRF data transmission, and in which the used bearer service is adapted for this data transmission.

What is claimed is:

1. A method in a communication system, which system (N, PLMN, PSTN, PDN) is arranged for transmitting information, requests (REQUEST), between a first mobile station (MS, MS1, MS2), a second mobile station .(MS, MS1, MS2) and a server (**3**, SERVER) connected to the communication system (N, PLMN, PSTN, PDN), wherein at least the first mobile station (MS, MS1, MS2) and at least the second mobile station (MS, MS1, MS2) comprise protocol means (**100**–**106**) for generating the request (REQUEST) and directing it to the communication system (N, PLMN, PSTN, PDN), which request (REQUEST) contains at least address information (URI, URL, URN) for identifying the destination of the request (REQUEST), and wherein at least the first mobile station (MS, MS1, MS2) comprises a first local resource (SC, **4**) coupled to the same, and in which method:

the request (REQUEST) is transmitted by the first mobile station (MS, MS1, MS2), the second mobile station (MS, MS1, MS2), or the server (**3**, SERVER),

characterized in that:

the address information (URI, URL, URN) of the request (REQUEST) is generated to identify said first local resource (SC, **4**), and

that the request (REQUEST) for said first local resource (SC, **4**) is generated and directed at least partly with said protocol means (**100**–**106**).

2. The method according to claim **1**, characterized in that:

the request (REQUEST) is generated in the second mobile station (MS, MS1, MS2), or in the server (**3**, SERVER),

that the first local resource (SC, **4**) coupled to the first mobile station (MS, MS1, MS2) is selected as the destination for the request (REQUEST), and

that the request (REQUEST) is transmitted at least partly by means of the communication system (N, PLMN, PSTN, PDN).

3. The method according to claim **1**, characterized in that:

the request (REQUEST) is generated in the first mobile station (MS, MS1, MS2), that the first local resource (SC, **4**) connected to the first mobile station (MS, MS1, MS2) is selected as the destination for the request (REQUEST), and

that the request (REQUEST) is transmitted and directed by with the protocol means (**100–106**) of the first mobile station (MS, MS1, MS2).

4. The method according to claim **1**, characterized in that the request (REQUEST) is generated at least partly by a local resource (SC, **4**) connected to the second mobile station (MS, MS1, MS2), or to the server (**3**, SERVER).

5. The method according to claim **1**, characterized in that address information (URI, URL, URN) identifying the local resource (SC, **4**) is entered into the application (**101**, BROWSER) by the user, to retrieve information from said local resource (SC, **4**), which application (**101**, BROWSER) is provided in the mobile station (MS, MS1, MS2) advantageously to present information for the user by means of a user interface (UI) of the mobile station (MS, MS1, MS2), and which application (**101**, BROWSER) is connected to the protocol means (**100–106**) to transmit said address information (URI, URL, URN).

6. The method according to claim **1**, characterized in that the request (REQUEST) is transmitted advantageously to receive an indication (INDICATION) from the local resource (SC, **4**), which indication (INDICATION) is directed at least partly with the help of said protocol means (**100–106**) to the mobile station (MS, MS1, MS2) or server (**3**, SERVER) which made the request (REQUEST).

7. The method according to claim **1**, characterized in that a WAP application protocol is applied to establish the protocol means (**100–106**).

8. The method according to claim **7**, characterized in that a connection is established at least from a WSP layer (**102**), a WTP layer (**103**), a WTLS layer (**104**), or a WDP layer (**105**) to an interface (**4**, SC INTERFACE) of said local resource (SC, **4**), which interface (**4**, SC INTERFACE) is connected to said local resource (SC, **4**), and which WSP layer (**105**), WTP layer (**103**), WTLS layer (**104**), or WDP layer (**105**) forms at least a part of the protocol means (**100–106**) of the WAP application protocol.

9. The method according to claim **1**, characterized in that a URI address, a URL address, or a URN name is used as address information.

10. The method according to claim **1**, characterized in that a smart card, such as a SIM card, connected to the mobile station (MS, MS1, MS2) is used as a local resource (SC, **4**).

11. A communication system, which system (N, PLMN, PSTN, PDN) is arranged for transmitting information requests (REQUEST) between a first mobile station (MS, MS1, MS2), a second mobile station (MS, MS1, MS2) and a server (**3**, SERVER) connected to the communication system (N, PLMN, PSTN, PDN), wherein these comprise protocol means (**100–106**) for generating a request (REQUEST) and directing it to the communication system (N, PLMN, PSTN, PDN), which request (REQUEST) contains at least address information (URI, URL, URN) for identifying the destination of the request (REQUEST), and

wherein at least the first mobile station (MS, MS1, MS2) comprises a first local resource (SC, **4**) connected thereto, characterized in that:

said first local resource (SC, **4**) is arranged to be identified by means of the address information (URI, URL, URN) of the request (REQUEST),

that the request (REQUEST) addressed to said first local resource (SC, **4**) is arranged to be generated and directed to said first local resource (SC, **4**) at least partly with the help of said protocol means (**100–106**), and that the communication system (N, PLMN, PSTN, PDN) is arranged to transmit the request (REQUEST) addressed to said first local resource (SC, **4**) by the second mobile station (MS, MS1, MS2) and/or the server (**3**, SERVER).

12. A wireless communication device comprising protocol means (**100–106**) for generating a request (REQUEST) and directing it to a communication system (N, PLMN, PSTN, PDN), which request (REQUEST) contains at least address information (URI, URI, URN) to identify the destination of the request (REQUEST), and which wireless communication device (MS, MS1, MS2) advantageously comprises a local resource (SC, **4**) connected thereto, characterized in that:

the local resource (SC, **4**) is arranged to be identified by means of the address information (URI, URL, URN) of the request (REQUEST), and

that said protocol means (**100–106**) are arranged for directing the request (REQUEST), addressed to said local resource (SC, **4**), to said local resource (SC, **4**).

13. The wireless communication device according to claim **12**, characterized in that said protocol means (**100–106**) are arranged for directing and transmitting a request (REQUEST) received by said wireless communication device (MS, MS1, MS2), a request (REQUEST) generated in said wireless communication device (MS, MS1, MS2), or a request (REQUEST) according to both these alternatives, to said local resource (SC, 4).

14. The wireless communication device according to claim **12**, characterized in that said protocol means (**100–106**) are arranged also to direct and transmit an indication (INDICATION) received as a response to the request (REQUEST) from the local resource (SC, **4**).

15. The wireless communication device according to claim **12**, characterized in that said protocol means (**100–106**) are also arranged for directing and transmitting the request (REQUEST) generated by said local resource (SC, **4**).

16. The wireless communication device according to claim **12**, characterized in that the local resource (SC, **4**) is a smart card, such as a SIM card.

17. The wireless communication device according to claim **12**, characterized in that the address information (URI, URL, URN) comprises a URI address, a URL address, or a URN name.

* * * * *

(E) Evidence for Claim 26 – Relied Upon

The following item (1) listed below is hereby entered as evidence relied upon by the Examiner as to grounds of rejection for claim 26, to be reviewed on appeal. Also listed for each item is where said evidence was entered into the record by the Examiner.

(1) Copy of US Patent Number 6,233,688 B1 ("Montenegro"). This evidence was entered into the record by the Examiner on page 11 at 13 of the Office Action mailed 08/10/2006.

Copies of all References follows.

//

US006233688B1

(12) **United States Patent**
Montenegro

(10) **Patent No.: US 6,233,688 B1**
(45) **Date of Patent: May 15, 2001**

(54) **REMOTE ACCESS FIREWALL TRAVERSAL URL**

(75) Inventor: **Gabriel Montenegro**, Fremont, CA (US)

(73) Assignee: **Sun Microsystems, Inc.**, Mountain View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/109,260**

(22) Filed: **Jun. 30, 1998**

(51) **Int. Cl.**^{7} **G06F 11/00**

(52) **U.S. Cl.** **713/201**

(58) **Field of Search** 713/201, 200, 713/202; 707/3, 4, 100, 102; 709/232, 237; 380/25, 29

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,818,019	*	6/1999	Valencia	395/200.57
5,822,539	*	10/1998	Van Hoff	395/200.66
5,944,823	*	8/1999	Jade et al.	713/201
5,950,195	*	9/1999	Stockwell et al.	707/4
5,987,611	*	11/1999	Freund	713/201
5,999,979	*	12/1999	Vellanki et al.	709/232
6,061,797	*	5/2000	Jade et al.	713/201
6,073,176	*	6/2000	Baindur et al.	709/227
6,088,796	*	7/2000	Cianfrocca et al.	713/152

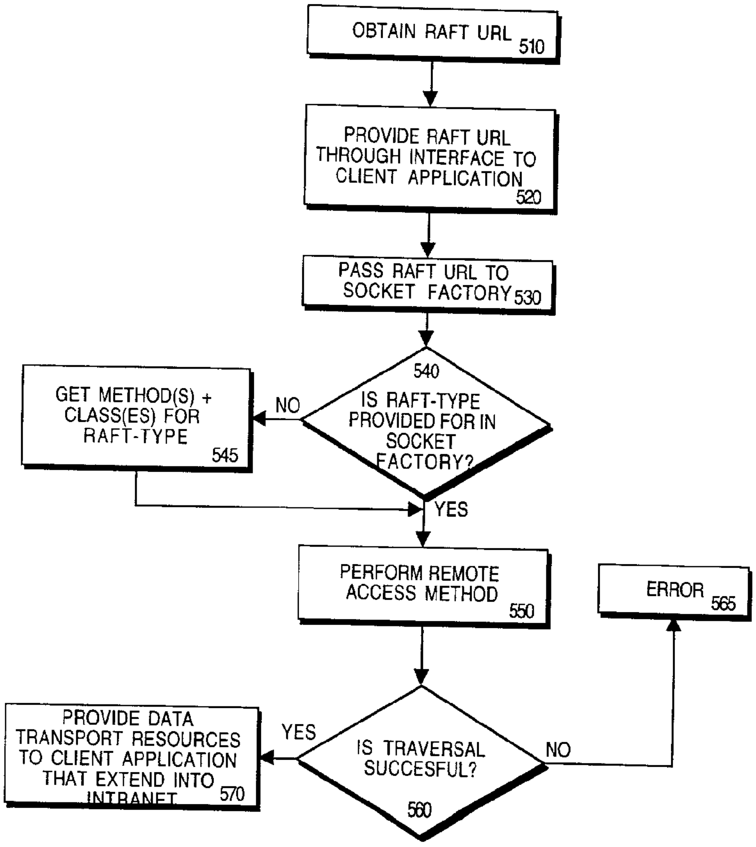
* cited by examiner

Primary Examiner—Nadeem Iqbal
(74) *Attorney, Agent, or Firm*—Blakely Sokoloff Taylor & Zafman

(57) **ABSTRACT**

The invention provides a generic naming scheme for remote access and firewall traversal in the form of a uniform resource locator (RAFT URL). The RAFT URL may be provided to any client, regardless of compatibility with the remote access/firewall traversal method, which then launches an operating environment code module. The operating environment code module performs the remote access/firewall traversal method and interacts with the operating environment to obtain data transport mechanisms. These mechanisms permit the client application to transact with private resources beyond the firewall. The remote access/firewall traversal procedure is made transparent to the client application, and thus, a wider array of client applications may be chosen for the data session with the resources beyond the firewall.

9 Claims, 5 Drawing Sheets


OBTAIN RAFT URL 510
PROVIDE RAFT URL THROUGH INTERFACE TO CLIENT APPLICATION 520
PASS RAFT URL TO SOCKET FACTORY 530
540
IS RAFT-TYPE PROVIDED FOR IN SOCKET FACTORY?
NO
GET METHOD(S) + CLASS(ES) FOR RAFT-TYPE 545
YES
PERFORM REMOTE ACCESS METHOD 550
ERROR 565
IS TRAVERSAL SUCCESFUL? 560
YES
PROVIDE DATA TRANSPORT RESOURCES TO CLIENT APPLICATION THAT EXTEND INTO INTRANET 570
NO

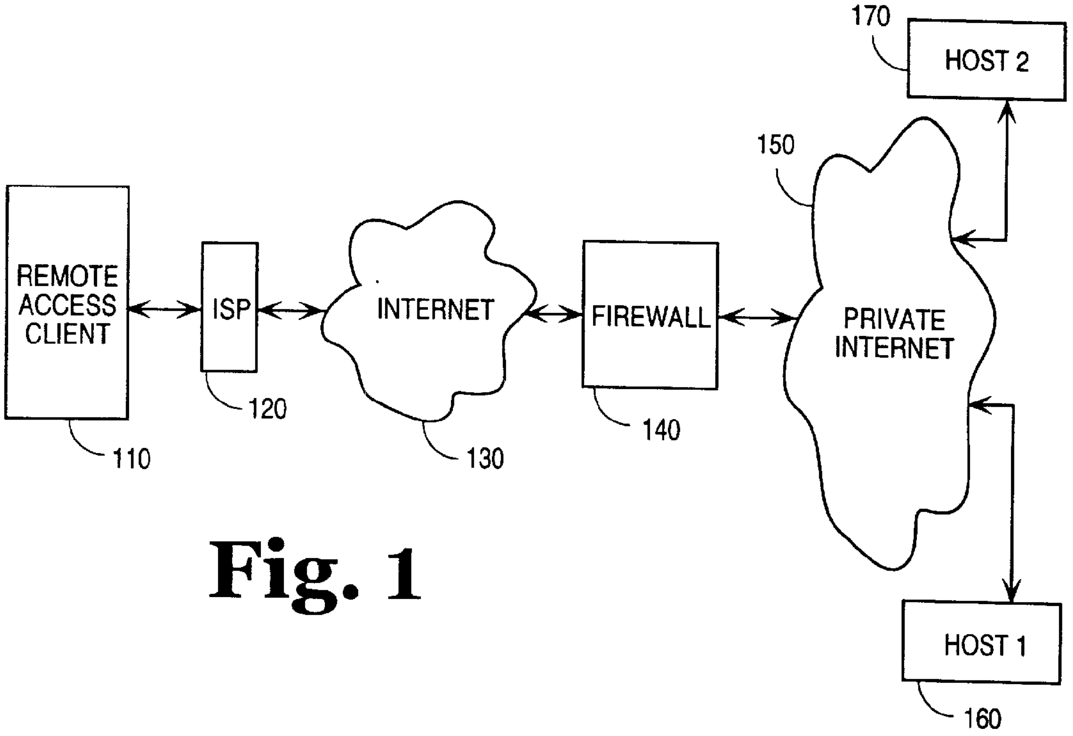

170
HOST 2
150
REMOTE ACCESS CLIENT
ISP
INTERNET
FIREWALL
PRIVATE INTERNET
120
110
130
140
HOST 1
160

Fig. 1

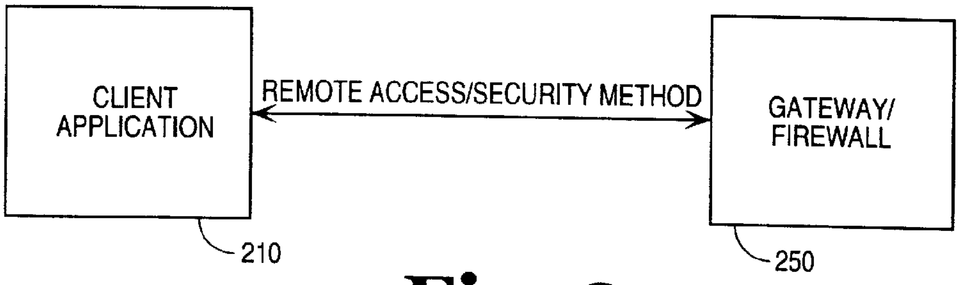

CLIENT APPLICATION
REMOTE ACCESS/SECURITY METHOD
GATEWAY/ FIREWALL
210
250

Fig. 2

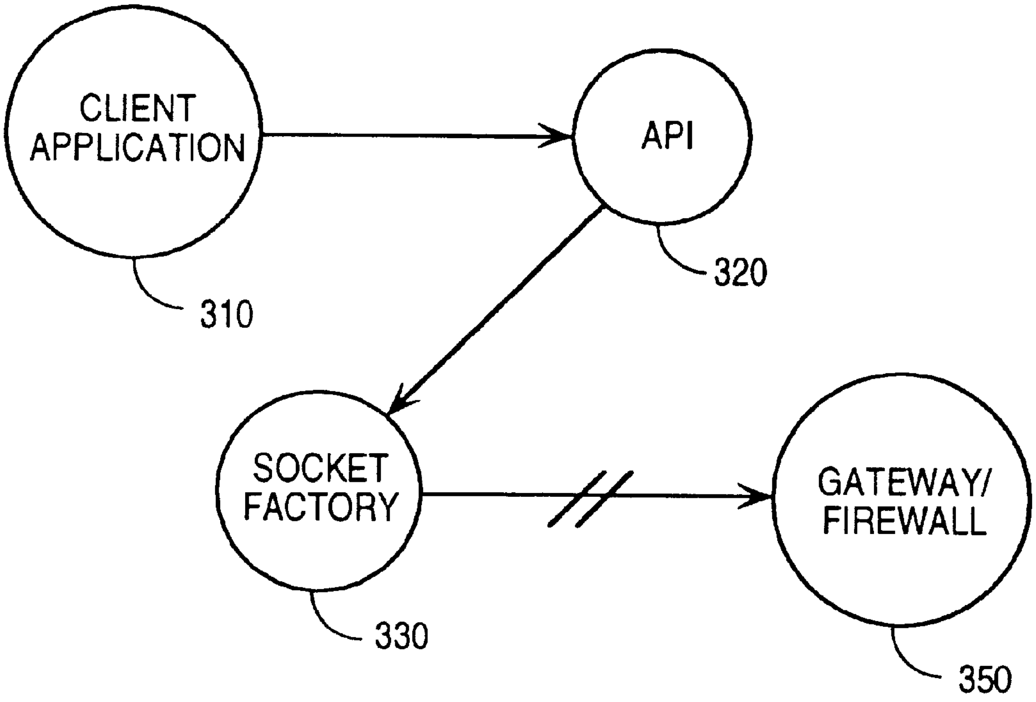

CLIENT APPLICATION
310
API
320
SOCKET FACTORY
330
GATEWAY/ FIREWALL
350

Fig. 3

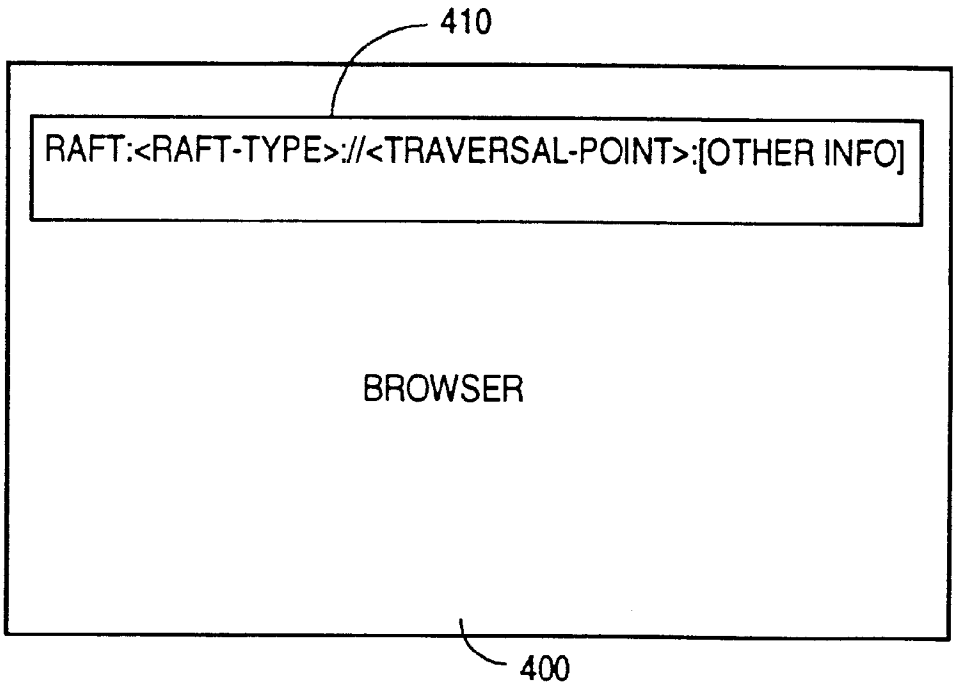

410
RAFT:<RAFT-TYPE>://<TRAVERSAL-POINT>:[OTHER INFO]
BROWSER
400

Fig. 4a

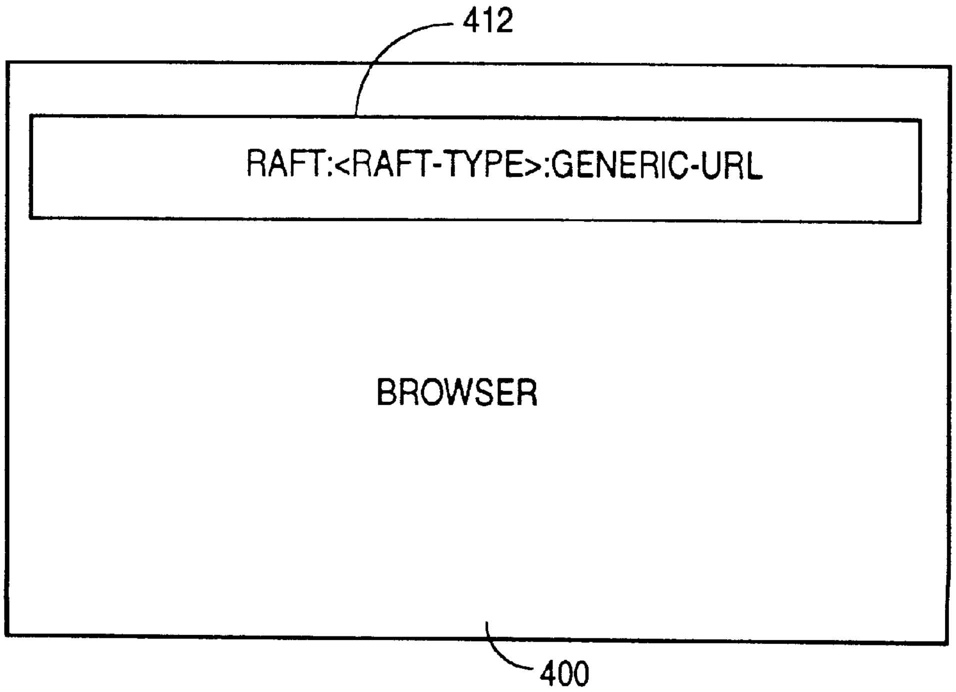

412
RAFT:<RAFT-TYPE>:GENERIC-URL
BROWSER
400

Fig. 4b

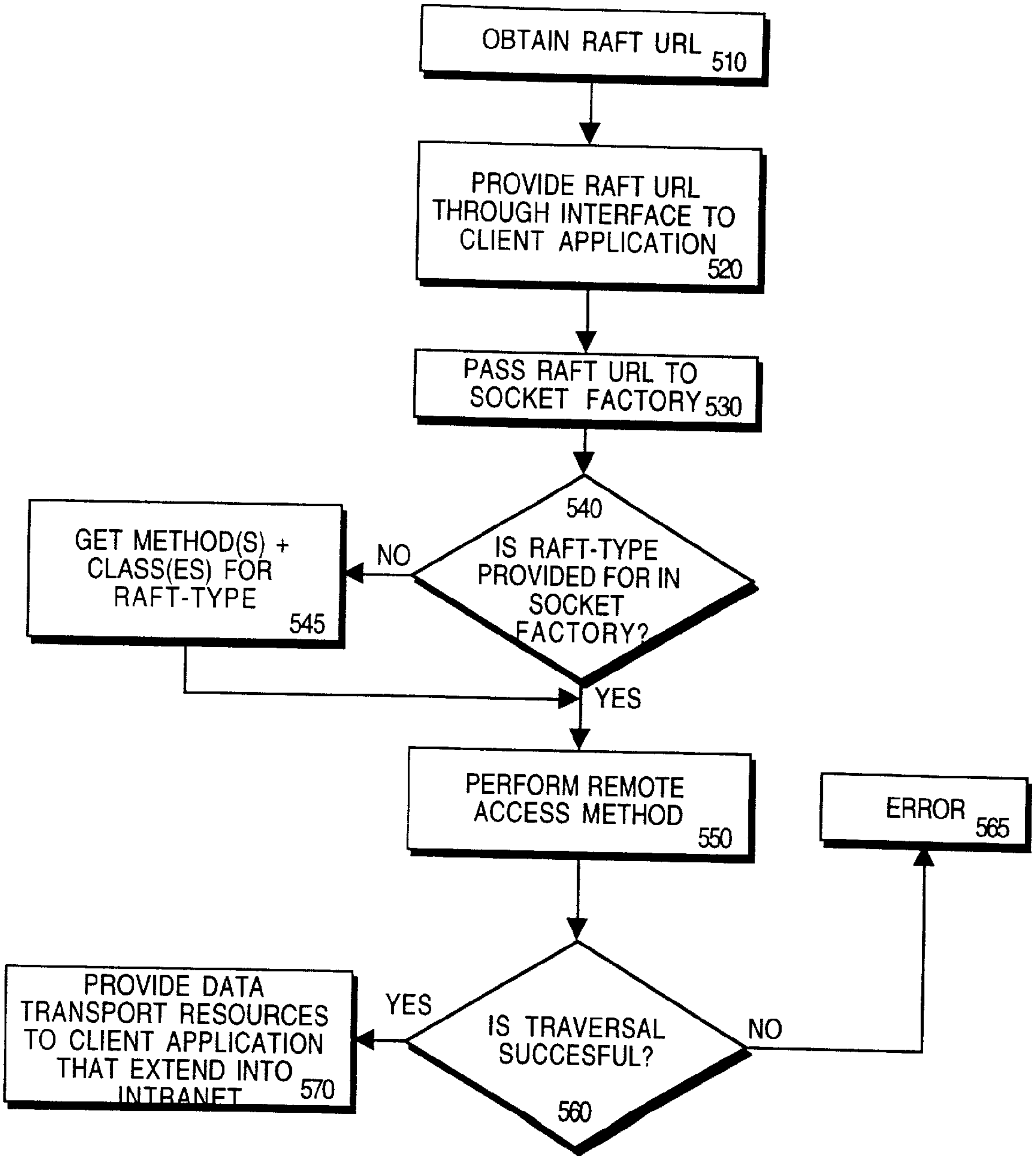

OBTAIN RAFT URL
510
PROVIDE RAFT URL THROUGH INTERFACE TO CLIENT APPLICATION
520
PASS RAFT URL TO SOCKET FACTORY
530
540
IS RAFT-TYPE PROVIDED FOR IN SOCKET FACTORY?
NO
GET METHOD(S) + CLASS(ES) FOR RAFT-TYPE
545
YES
PERFORM REMOTE ACCESS METHOD
550
ERROR
565
IS TRAVERSAL SUCCESFUL?
560
YES
NO
PROVIDE DATA TRANSPORT RESOURCES TO CLIENT APPLICATION THAT EXTEND INTO INTRANET
570

Fig. 5

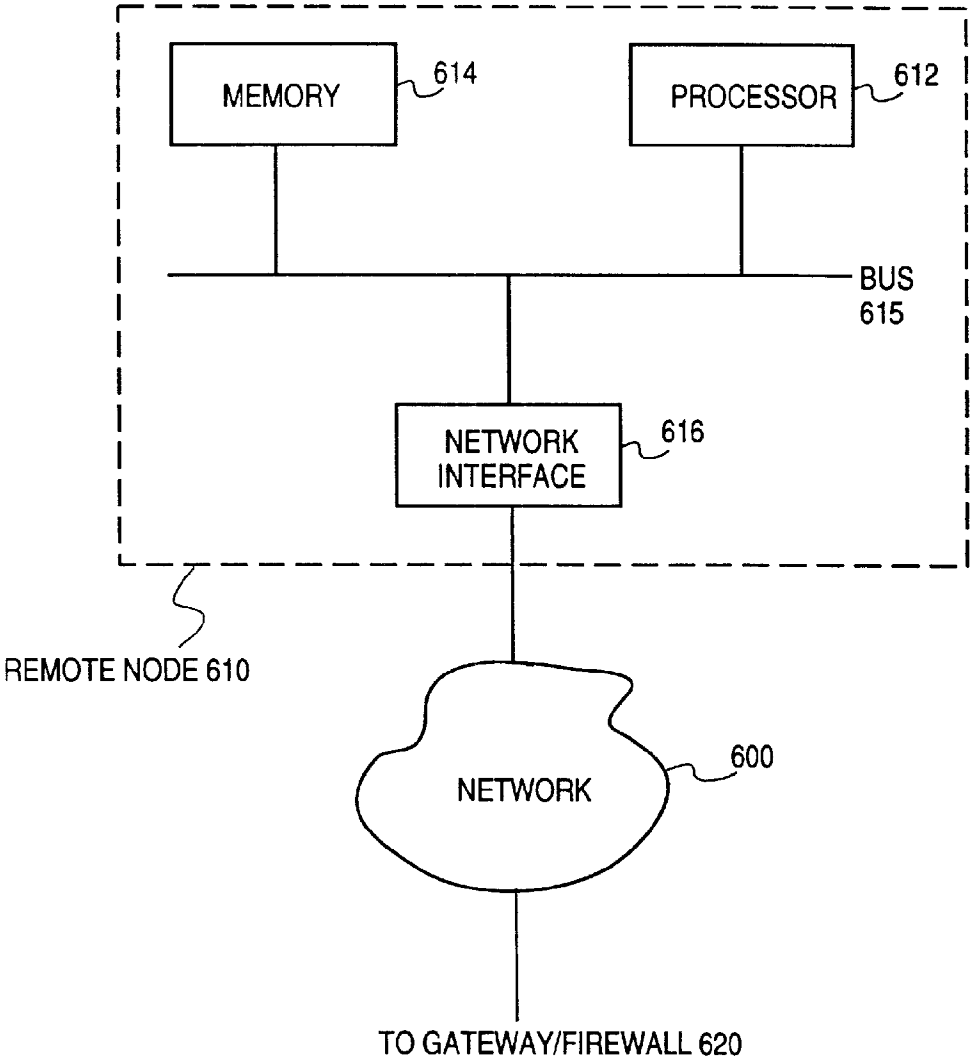

MEMORY
614
PROCESSOR
612
BUS
615
NETWORK
INTERFACE
616
REMOTE NODE 610
NETWORK
600
TO GATEWAY/FIREWALL 620

Fig. 6

REMOTE ACCESS FIREWALL TRAVERSAL URL

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention relates generally to the field of computer networking. More specifically, the invention relates to protocols and conventions for computer inter-networking.

2. Description of the Related Art

From within a corporate "intranet" network or a shared private network, the methods and protocols for local area access to computers, devices and data resources within the network is well defined and somewhat uniform, under the control of the administrators of those networks. When users attempt to gain access to those same devices, computers and resources from outside the network, such access is referred to as "remote access". In the past, the most popular physical topology for remote access is a direct dial into a modem bank, which may be at the corporate site or provided for by an ISP (Internet Service Provider). However, this topology impose a heavy administrative burden and monetary cost especially when remote access is attempted through long distance or international toll calls. Thus, there has been a recent trend to provide remote access through an Internet connection. With Internet remote access, the IP (Internet Protocol) connection can be obtained first using any available method, and thus the intranet does not need to maintain a direct physical access point such as a dial-in modem bank. Once a user is "on the Internet" (has achieved an IP connection), a multitude of different protocols and services (limited by the connectivity features of the intranet) can be used on the user's "client" to gain remote access into the intranet. In order to gain remote access, the client must pass the intermediary step, in most cases, of traversing a firewall. The traversal of the firewall can be achieved by using gateway specific software, SSL (Secure Sockets Layer) mechanisms and so on.

Specific client software must have support and awareness of specific firewall traversal methods, and thus generic client software cannot be utilized to penetrate the intranet. For example, a client application such as Netscape™ may not be able to traverse the firewall since it lacks the means with which to express entry parameters to "support" the private intranet's firewall scheme. Thus, users are often limited to using software that specifically understands and communicates with the intranet. This restricts the choice of client software greatly such that only a limited set of client applications out of all the multitude of programs available can be used when accessing that private intranet.

These schemes typically tie the firewall access mechanism to the application, instead of making it transparent by placing it inside the underlying networking support. There is a need for general naming mechanism in order to separate application from firewall traversal mechanisms. Furthermore, the firewall has no standard format to download traversal configuration into the client after authentication.

Thus, there is a need for a generic scheme for allowing client applications to be transparent to the remote access and firewall traversal procedure. The scheme should permit any type of remote access/firewall traversal and security method/protocol to be recognized and operated independent of the client application.

SUMMARY OF THE INVENTION

The invention provides a generic naming scheme for remote access and firewall traversal in the form of a uniform resource locator (RAFT URL). The RAFT URL may be provided to any client application, regardless of compatibility with the remote access/firewall traversal method, which then launches another executable module. The executable module performs the remote access/firewall traversal method and interacts with the operating environment to obtain data transport mechanisms. These mechanisms permit the client application to transact with private resources beyond the firewall. The remote access/firewall traversal procedure is made transparent to the client application, and thus, a wider array of client applications may be chosen for the data session with the resources beyond the firewall.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects, features and advantages of the method and apparatus for the present invention will be apparent from the following description in which:

FIG. **1** illustrates the topology of remote access to a private intranet.

FIG. **2** illustrates a traditional remote access scheme.

FIG. **3** is a logical diagram of one embodiment of the invention.

FIG. **4***a* illustrates a RAFT URL according to one embodiment of the invention.

FIG. **4***b* illustrates a RAFT URL according to another embodiment of the invention.

FIG. **5** is a flow diagram of RAFT URL processing according to one embodiment of the invention.

FIG. **6** is a diagram of one embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

Referring to the figures, exemplary embodiments of the invention will now be described. The exemplary embodiments are provided to illustrate aspects of the invention and should not be construed as limiting the scope of the invention. The exemplary embodiments are primarily described with reference to block diagrams or flowcharts. As to the flowcharts, each block within the flowcharts represents both a method step and an apparatus element for performing the method step. Depending upon the implementation, the corresponding apparatus element may be configured in hardware, software, firmware or combinations thereof.

FIG. **1** illustrates the topology of remote access to a private intranet.

Internet-based remote access may involve many different entities which intervene between a remote access client **110** and hosts such as host **160** and host **170**. Remote access client **110** may consist of a computer system or PDA (Personal Digital Assistant) or other information device that has the capability of being connected to a network. Remote access client **110** may also execute software configured to enable the access of services such as FTP (File Transfer Protocol), POP (Post Office Protocol), HTTP (HyperText Transfer Protocol), etc. One way of "getting on the Internet" to commence the remote access session with one or more hosts **160** and **170** is to dial-up to an ISP (Internet Service Provider) whose internal routers and switches provide a TCP/IP (Transport Control protocol/Internet Protocol) connection to remote access client **110**. With the "Internet connection thus obtained, the remote access client **110** can reach any computer connected to the "Internet" **130** which is well-known in the art. The ISP dial-up method of Internet access described above is merely exemplary of many ways a connection to the Internet may be obtained.

Once on the Internet **130**, remote access client **110** is free to access services provided by any computers or networks connected to the Internet **130** such as an HTTP service (i.e., a web site). However, the below description restricts itself to using the Internet to gain remote access into a private network or intranet **150** by means of the Internet. Once remote access into the intranet **150** is established, remote access client **110**, services provided for within the intranet **150** may be accessed. What distinguishes the intranet **150** from an ordinary web server or FTP server on a more public network is the security and isolation that can be provided by a gateway or firewall **140**. One function of firewall **140** is to prevent unauthorized access into the intranet by users/computers connected to the Internet **130**.

To control and configure access to the intranet **150** through firewall **140** many various firewall protections or security schemes have been developed such as IP security schemes using SKIP (Simple Key Management for Internet Protocols), or ISAKMP (Internet Security Association and Key Management Protocol), SSL (Secure Sockets layer), etc. Many of these schemes are conflicting and standardization has not been successfully achieved. One reason for the failure of standardization is the nature of remote access—it is intended for a specific often closed set of users. For instance, a company X may desire that only certain key employees have remote access to the intranet of X. In that case, company X will choose whatever method of remote access security is easy to implement or whatever method is decided on as best for the type of information served. Since the choice of remote access security methodology is isolated to the company implementing it, standardization is difficult. In a more public network such as the Internet standardization is easy to achieve since the many nodes of the network desire compatibility with the other.

Lack of standardization limits the remote access client's **110** choice of remote access security methods when using software to access services provided by the intranet.

The invention in its various embodiments permits the use of a wider range of client software to access the intranet **150**. A naming scheme is provided which when used by client software will cause the client **110** to negotiate remote access. The naming convention is generalized so that any of the conventional security methods can be adequately identified. In one embodiment, the client software parses a RAFT URL (Remote Access Firewall Traversal Uniform Resource Locator) and executes code that extends the ability of the underlying operating system of remote access client **110** to negotiate access according to the RAFT URL. The invention in certain embodiments makes transparent the process of firewall traversal. This transparency will allow the use of application software that is not restricted by or concerned with the remote access security and traversal method.

Currently, there is no transparency between the procedure of firewall traversal for remote access and the client application software used to thereafter access the services provided by the intranet beyond the firewall, if the traversal method uses application layer traversal mechanisms. FIG. **2** illustrates this prior art model of application dependency upon the remote access security method. A client application **210** is executed on a system somewhere outside in a topology sense from the firewall **250** and intranet.

In traversal methods like httpstunneling, client application **210** must be an application specifically aware and capable of the remote access security and traversal method. The particulars of this method are predefined by the gateway **250** and cannot be modified. As such, if an application such as a web browser does not have built-in support for the remote access method, it will be unable to gain remote access. The process of attaining remote access and/or traversing a firewall is thus, within the purview of the client application in application layer transversal mechanisms.

The RAFT URL mechanism of the invention provides benefits application or session layer traversal mechanisms that imply client software modifications. The RAFT URL provides a universal language to denote what these modifications are. It promotes the standardized use of these mechanisms as a consistent set of functions that become a service to client applications rather than an inherent part of them.

For firewalls in general, RAFT URLs provides a universal language that aids in unifying firewall technology. Currently, some firewalls are based on session layer or application layer mechanisms (application layer gateways or ALGs), whereas other firewalls operate at lower layers, in a much more transparent (to applications) fashion, i.e., IPSEC firewalls, network layer tunneling firewalls. RAFT allows these firewall technologies to be treated in a similar fashion, and to allow their integration and tracking within one single mechanism.

FIG. **3** is a logical diagram of one embodiment of the invention.

Unlike the prior art model of FIG. **2**, the client application **310** does not have the responsibility of directly securing remote access. Rather, the client application **310** makes use of a socket factory **320** (Application Program Interface) to access a system resource such as sockets. The socket factory **320** negotiates security and access to gateway/firewall **350**. The socket factory is configured to understand the naming convention (RAFT URL) and then initiate steps to obtain remote access, which includes the negotiation of security protocols defined by the RAFT URL. The RAFT URL is input either by user or by preference setting to the client application **310**. The socket factory recognizes the RAFT URL (see FIG. **5**) and configures itself **330** to communicate with gateway/firewall **350**.

This socket factory **320** can be made available to any other client application as well, such as a client application **315** or **312**. Like client application **310**, client applications **312** do not need to be compliant or compatible with the firewall security method. When client applications **312** or **315** obtain sockets via the socket factory, these applications will inherit the same behavior as provided in the sockets to transmit/receive information from gateway/firewall **350**. Once the socket factory has successfully gained access beyond the firewall, the sockets derived from the socket factory **330** provide presentation layer data transport mechanisms to client application **310**.

Client application **310** (or **312** or **315**) would be unaware, except for the obtaining of behavior from the sockets of the firewall traversal. Once client application **310** has obtained the right to transact via the sockets mechanism to gateway/firewall **350**, other applications **312** and **315** can use sockets in a similar fashion to communicate beyond the firewall **350**. The traversing of the firewall is divorced from the client application **310** and made the responsibility of the socket factory **320**. In so doing, traversal of the firewall and its security method is made transparent to client applications **310**, **312** and **315**. Unlike the prior art of FIG. **2**, this permits the client application to be less specific to the firewall.

FIG. **4***a* illustrates a RAFT URL according to one embodiment of the invention.

The RAFT URL is a naming scheme generic to the various firewall traversal and remote access security methods. Whatever method the particular firewall designates, the RAFT URL has a structure that can contain identifiers for the underlying implementation (socket factory in the Java case) to handle its negotiation.

FIG. **4***a* shows a system application **400** like one used to configure the socket factory which accepts as input or can store as preference a RAFT URL **410**. Unlike an ordinary URL, the RAFT URL does not point to a data object or data creator. Rather, it designates the means to negotiate firewall access which includes specifying security methods either implicitly or explicitly. In one embodiment, the RAFT URL **410** has the following components:

"raft"—indicates the identifying information to follow are handles to configure remote access. Unlike "http:" or "ftp:" of an ordinary URL which identifies a data transfer protocol or service, the "RAFT:" designation can serve to launch or call the remote access mechanism.

"raft-type"—designates the particular name given to a specific firewall traversal or remote access method. For instance, the use of layer **3** tunneling with SKIP (Simple Key-Management for Internet Protocols) or tunneling through the firewall using SSL.

"traversal point"—indicates the IP address, domain name or other location of the firewall, gateway, or remote access server with which the client system must negotiate access, the traversal point will be known to an authorized user or can be provided through some other means through authentication.

"other-info"—indicates a security scheme specific initialization string such as a password, user name or even a secondary security mechanism. The parameter is optional with its format defined wholly by the scheme and firewall's policy.

"generic-URL"—allows regular URL's of the "http:" or "ftp:" variety.

FIG. **4***b* illustrates a RAFT URL according to another embodiment of the invention.

Like FIG. **4***a*, both the designation of "raft:" and a raft-type are provided. Instead of a traversal point, a generic-URL parameter terminating it. Such an embodiment for the RAFT URL may be used in https or secure http protocols, where the "https" URL designates the entry point through the firewall.

Implementation for Specific Schemes

Described in a listing of desirable implementations of the RAFT URL for specific remote access and firewall traversal schemes. The RAFT URL concept presented in its various embodiments permits any type of scheme to be designated.

1. SSL Tunneling

The RAFT URL for SSL tunneling would take the form "raft:sslt:https://x" where x may designate a host port or server address, directory path and search/cgi (common gateway interface) or file name parameters.

2. Mobile IP Firewall Traversal

The RAFT URL for firewall traversal by mobile IP (Internet Protocol) would take the form "raft:mip://traversal-point" with an optional terminating string ";type=y", where y refers to either the "SKIP" (Simple Key Management for Internet Protocols) security protocol or IP security protocols.

3. Remote Access Using TSP

The RAFT URL for remote access using TSP (Tunneling Set-up Protocol) takes the form "raft:tsp://x" where x includes a traversal point and optional ";type=y" terminating parameter. Y may be either "ipsec" or "skip" as defined for Mobile IP access in part 2 above.

For instance, consider the following RAFT URL" "raft:sslt:https://firewall.foo.com/access.html". This RAFT URL indicates that remote access through the firewall named/addressed as "firewall.foo.com" is to be achieved using secure http by processing the page "access.html" which may be an interactive login page where a user enters login information such as a user name and password for further entry beyond the firewall.

The RAFT URL is useful due to its versatility; any type of remote access may be designated and specified. A second feature of the invention lies in the processing of the RAFT URL in a manner transparent to the client data application.

FIG. **5** is a flow diagram of RAFT URL processing according to one embodiment of the invention.

The first step in creating a transparency between the firewall traversal for remote access and the client application is to obtain the appropriate RAFT URL (step **510**). The discovery of the specific RAFT URL to use is not a subject of the essential invention. Obtaining a RAFT URL may be achieved in several ways: (1) obtaining it in person from a system administrator, (2) visiting a special web page where an authenticated user may retrieve the appropriate RAFT URL from the firewall, (3) querying a directory service such as LDAP (Lightweight Directory Access Protocol) or (4) may be preconfigured into the client application or system. The appropriate RAFT URL will designate parameters allowing the client system to get private intranet resources through its data transport mechanisms (IP stack, sockets, etc.).

Assuming that the RAFT URL is obtained, it is then provided through some interface to the client application (step **520**). This interface may be URL data entry dialog of a browser or be chosen from a menu by the user. The RAFT URL may already be provided to the client application by being set-up in a preference manager for the client application or in an operating system registry intended to service that client application. When so provided, the RAFT URL is passed to the socket factory (in Java) (step **530**) that will execute methods needed to perform the firewall traversal procedure.

If the specific "raft-type" (type of firewall traversal and/or remote access security, see above) is not provided for in the socket factory, by way of methods, functions, classes and/or code that can be executed, then the required methods, functions, classes, code, etc. must be obtained. In that case, the firewall traversal procedure first gets the needed methods, classes, etc. for the raft-type. These methods, classes, etc. may be obtained from the firewall itself and then loaded as an API (Application Program Interface) or mobile code, such as Java applet, onto the client system. If the required methods, classes, etc., to undergo processing of the "raft-type" is available or made available then the remote access method is performed (step **550**). The remote access and/or firewall traversal method, as specified by the "raft-type" parameter of the RAFT URL, need not be known or compatible with the client application(s) ultimately handling the data transactions with resources beyond the firewall. This responsibility and restriction is removed to the socket factory.

If the traversal or remote access is successful (checked at step **560**), the data transport resources are provided to the client application that extend into the intranet (step **570**). The data transport resources may be sockets, a TCP/IP stack or other presentation/transport layer mechanisms which facilitate data transport between the client system and resources such as servers existing in the intranet beyond the firewall. Access is regulated by and monitored by the firewall. The interaction between the API/applet and the obtained data transport resource is dependent upon platform and operating system and will vary from system to system.

For example, consider a mobile network computer system based upon Java. Such a computer system typically uses a socket factory to create network data transport resources (called sockets). These sockets allow applications to transact data over a network such as the Internet. Currently, firewall traversal and remote access security operates above this transport layer on the application layer. As a consequence, the choice of client applications restricted when transacting data within an intranet beyond the firewall since compatibility is required. SSL attempts to generalize secure access but does so on the application level and thus, the client application too must be SSL compatible. Further, if SSL is not offered by the firewall as the secure access method, then SSL fails to be a general solution which will be adequate for every firewall traversal and remote access security situation. In the Java based mobile computer system, the invention, in one embodiment, would operate to set the socket factory according to parameters in the RAFT URL. Client applications that use the standard network resources Java application (known as java.net) can also be divorced from having to include the methods, functions, classes, etc. needed to perform remote access in accordance with the RAFT URL. Rather, a system applet would parse the RAFT URL and set the socket factory by adding methods, functions, classes (if not already present) and then allow the socket factory to handle the firewall traversal and remote access. If a client application wishes directly control its remote access security scheme, it may directly add the required behavior (methods, functions, classes, etc.) to configure its use of the socket factory provided for in the operating system. The actual negotiation of remote access or firewall traversal is made transparent to the application layer, which takes advantage of data transport resources after firewall traversal is successfully complete.

The above example refers to a Java-based mobile computing platform. However, the invention, in its various embodiments, is not limited to any platform or operating environment. An application or other network data transport resource can be provided with the means to accept and parse the RAFT URL and then carry out the identified remote access procedure. If the remote access behavior is not available to the operating system, it may be obtained automatically by the application.

FIG. **6** is a diagram of one embodiment of the invention.

A RAFT URL mechanism would allow a remote node **610** to traverse into gateway/firewall. Remote node **610** is illustrative of a computer system or information device that desires remote access to a private or other network that lies beyond the firewall. Remote node **610** may be connected to gateway/firewall via a network **600** such as the Internet.

Remote node **610** may be composed of a variety of devices, including a memory **614** and a processor **612** coupled directly to each other and through a bus **615**. Bus **615** may also attach a network interface card **616** to both memory **614** and processor **612**. Network interface card **616** connects the remote **610** to network **600** and allows remote node **610** to transact data to and from network **600** and consequently, to and from other nodes that may be connected to network **600** such as the gateway/firewall. A RAFT URL that identifies the firewall and its security access scheme may be provided to remote node **610** in a variety of ways, including transmission over network **600**. Once the appropriate RAFT URL is provided to remote node **610** the RAFT URL is passed to a socket factory generated by some application running in memory **614** and executed by processor **612**. The processor **612** is configured (by instructions provided thereto) to parse the RAFT URL and initiate the appropriate client events on remote node **610** for traversal of gateway/firewall **620**. Once remote access is granted to remote node **610**, it is free to transact data with resources beyond the firewall **620**. This data transaction is handled by network interface **616** which may modify data packets with any security-related headers or encapsulation demanded by the RAFT URL's designation.

The exemplary embodiments described herein are provided merely to illustrate the principles of the invention and should not be construed as limiting the scope of the invention. Rather, the principles of the invention may be applied to a wide range of systems to achieve the advantages described herein and to achieve other advantages or to satisfy other objectives as well.

What is claimed is:

1. A method of remote access comprising:

providing a remote access firewall traversal (RAFT) uniform resource locator (URL) to a client, said RAFT URL indicating a mode of remote access through a firewall; and

configuring said client to negotiate access to a private resource protected by said firewall based on parameters specified by said RAFT URL that allow said client to access said private resource through its data transport mechanisms.

2. A method according to claim **1** wherein the step of recognizing a RAFT URL includes the steps of:

identifying a RAFT service;

identifying a RAFT type, said RAFT type denoting a specific remote access method.

3. A method according to claim **2** wherein the step of recognizing a RAFT URL further includes the steps of:

identifying a generic uniform resource locator.

4. A method according to claim **2** wherein the step of recognizing a RAFT URL further includes the steps of:

identifying a traversal point.

5. A method according to claim **4** wherein the step of recognizing a RAFT URL further includes the steps of:

identifying a scheme-specific initialization string.

6. A method according to claim **1** wherein the step of configuring includes the steps of:

building sockets from a socket factory in accordance with said RAFT URL; and

passing the behavior of said sockets from said socket factory to application on said client, said application able to traverse said firewall with the appropriate method.

7. A method according to claim **6** further comprising the step of:

obtaining methods and classes for a raft-type specified in said RAFT URL if said raft-type is not provided for by said socket factory.

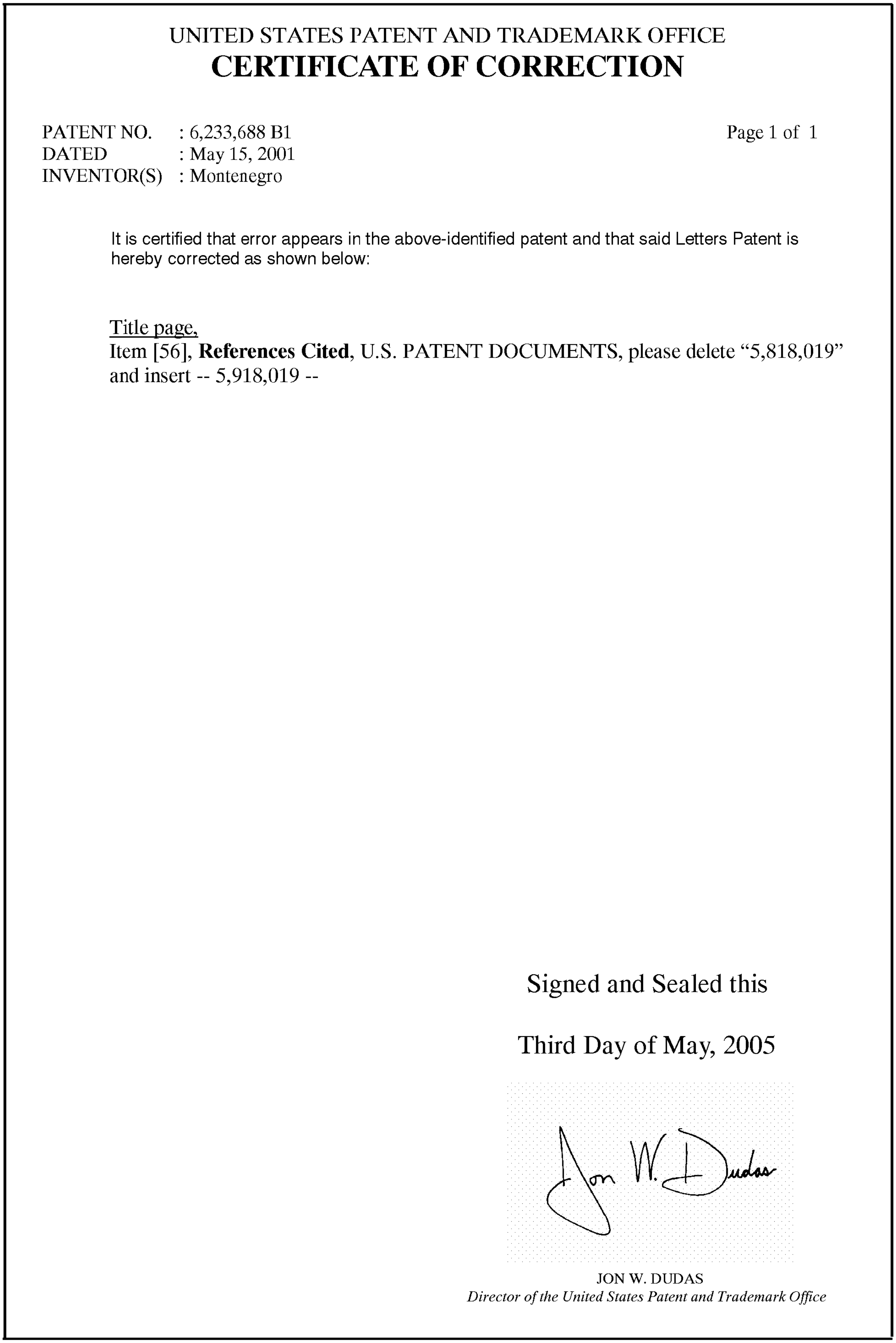
8. A method according to claim **2** further comprising the step of:

performing said specific remote access method, and if said performing is successful, providing data transport resources that extends to said private resource to said client.

9. A computer readable medium comprising:

instructions when executed by a processor cause said processor to provide remote access firewall traversal, said instructions including a remote access firewall traversal (RAFT) uniform resource locator (URL, said RAFT URL indicating a mode of remote access through a firewall.

* * * * *



UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,233,688 B1 Page 1 of 1
DATED : May 15, 2001
INVENTOR(S) : Montenegro

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title page,
Item [56], **References Cited**, U.S. PATENT DOCUMENTS, please delete "5,818,019" and insert -- 5,918,019 --

Signed and Sealed this

Third Day of May, 2005

Jon W. Dudas

JON W. DUDAS
Director of the United States Patent and Trademark Office

(F) Evidence for Claims 1-26 – Entered by Examiner

The following items (1) – (17) listed below are hereby entered as evidence entered by the Examiner. Also listed for each item is where said evidence was entered into the record by the Examiner.

(1) Copy of US Patent Number 5,987,611 ("Freund"). This evidence was entered into the record by the Examiner on page 4 at 6 of the Office Action mailed 01/24/2006 and on page 1 of 1 Notice of References Cited line A.

(2) Copy of US Patent Number 6,950,947 B1 ("Purtell et al"). This evidence was entered into the record by the Examiner on page 2 at 5 of the Office Action mailed 01/24/2006 and on page 1 of 1 Notice of References Cited line B.

(3) Copy of US Patent Number 5,852,717 ("Bhide et al"). This evidence was entered into the record by the Examiner on page 5 at 7 of the Office Action mailed 01/24/2006; on page 5 at 7 of the Office Action mailed 05/05/2005 and on page 1 of 1 Notice of References Cited line A.

(4) Copy of US Patent Number 6,609,154 B1 ("Fuh et al"). This evidence was entered into the record by the Examiner on page 6 at 8 of the Office Action mailed 01/24/2006, on page 6 at 8 of the Office Action mailed 05/05/2005, and on page 1 of 1 Notice of References Cited line E.

(5) Copy of US Patent Number 6,549,773 B1 ("Linden et al"). This evidence was entered into the record by the Examiner on page 8 at 10 of the Office Action mailed 01/24/2006, on page 8 at 10 of the Office Action mailed 05/05/2005, and on page 1 of 1 Notice of References Cited line D.

(6) Copy of US Patent Number 6,233,688 B1 ("Montenegro"). This evidence was entered into the record by the Examiner on page 10 at 12 of the Office Action mailed 01/24/2006, on page 10 at 12 of the Office Action mailed 05/05/2005, and on page 1 of 1 Notice of References Cited line B.

(7) Copy of US Patent Number 6,484,206 B2 ("Crump et al"). This evidence was entered into the record by the Examiner on page 3 at 5 of the Office Action mailed 05/05/2005 and on page 1 of 1 Notice of References Cited line C.

(8) Copy of US Patent Number 6,854,063 B1 ("Qu et al"). This evidence was entered into the record by the Examiner on page 4 at 6 of the Office Action mailed 05/05/2005 and on page 1 of 1 Notice of References Cited line F.

(9) Copy of U.S. Patent 6,550,012 B1 ("Villa et al."). This evidence was entered into the record by the Examiner on on page 1 of 1 Notice of References Cited line A of the Office Action mailed 08/10/2004.

(10) Copy of U.S. Patent 5,950,195 A ("Stockwell et al."). This evidence was entered into the record by the Examiner on on page 1 of 1 Notice of References Cited line B of the Office Action mailed 08/10/2004.

(11) Copy of U.S. Patent 5,983,350 A ("Minear et al."). This evidence was entered into the record by the Examiner on on page 1 of 1 Notice of References Cited line C of the Office Action mailed 08/10/2004.

(12) Copy of U.S. Patent 6,061,797 A ("Jade et al."). This evidence was entered into the record by the Examiner on on page 1 of 1 Notice of References Cited line D of the Office Action mailed 08/10/2004.

(13) Copy of U.S. Patent 6,651,174 B1 ("Nagaoka et al."). This evidence was entered into the record by the Examiner on on page 1 of 1 Notice of References Cited line E of the Office Action mailed 08/10/2004.

(14) Copy of U.S. Patent Application 2002/0078198 A1 ("Buchbinder et al."). This evidence was entered into the record by the Examiner on on page 1 of 1 Notice of References Cited line F of the Office Action mailed 08/10/2004.

(15) Copy of U.S. Patent 6,219,786 B1 ("Cunningham et al."). This evidence was entered into the record by the Examiner on page 8 paragraph 3 of the Office Action mailed 08/10/2004 and on page 1 of 1 Notice of References Cited line G.

(16) Copy of WO 98/34385 A1 ("Vellanki et al."). This evidence was entered into the record by the Examiner on page 3 paragraph 2 of the Office Action mailed 08/10/2004

(17) Copy of U.S. Patent 6,044,401 A ("Harvey"). This evidence was entered into the record by the Examiner on page 5 paragraph 4 of the Office Action mailed 08/10/2004

Copies of all References follows.

//

US005987611A

United States Patent [19]

Freund

[11] **Patent Number: 5,987,611**

[45] **Date of Patent: Nov. 16, 1999**

[54] **SYSTEM AND METHODOLOGY FOR MANAGING INTERNET ACCESS ON A PER APPLICATION BASIS FOR CLIENT COMPUTERS CONNECTED TO THE INTERNET**

[75] Inventor: **Gregor Freund**, San Francisco, Calif.

[73] Assignee: **Zone Labs, Inc.**, San Francisco, Calif.

[21] Appl. No.: **08/851,777**

[22] Filed: **May 6, 1997**

Related U.S. Application Data

[60] Provisional application No. 60/033,975, Dec. 31, 1996.

[51] **Int. Cl.**^{6} **G06F 13/00**

[52] **U.S. Cl.** **713/201**

[58] **Field of Search** 395/187.01, 186; 364/222.5, 286.4, 286.5; 711/163; 707/9, 10, 203; 713/200, 201

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,914,586	4/1990	Swinehart et al.	364/200
5,475,817	12/1995	Waldo et al.	395/650
5,586,260	12/1996	Hu	395/200.2
5,623,601	4/1997	Vu	395/187.01
5,764,887	6/1998	Kells et al.	395/186
5,815,574	9/1998	Fortinsky	380/25
5,828,833	10/1998	Belville et al.	395/187.01
5,832,211	11/1998	Blakley, III et al.	395/188.01
5,838,903	11/1998	Blakely, III et al.	395/188.01
5,857,191	1/1999	Blackwell, Jr. et al.	707/10
5,864,665	1/1999	Tran	395/187.01
5,875,296	2/1999	Shi et al.	395/188.01
5,881,230	3/1999	Christensen et al.	395/200.33

OTHER PUBLICATIONS

Mullender, "Distributed Systems", Second Edition, ACM Press New York, Addison–Wesley, pp. 3. 12–13, 543–578, Dec. 1993.

ORFALI et al., "Essential Client/Server Survival Guide", Van Nostrand Reinhold, pp. 153–154, Dec. 1994.

Postel, J., "RFC 821—Simple Mail Transfer Protocol," Information Science Institute, University of Southern California, Aug. 1982, pp. 1–68.

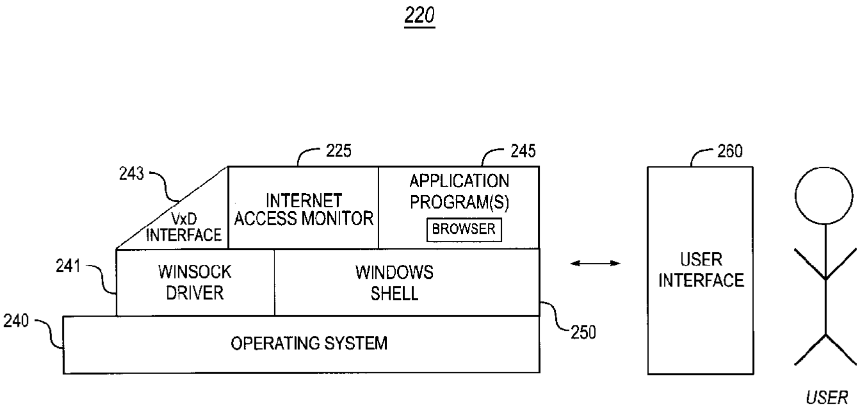
(List continued on next page.)

Primary Examiner—Robert W. Beausoliel, Jr.
Assistant Examiner—Stephen C. Elmore
Attorney, Agent, or Firm—John A. Smart

[57] **ABSTRACT**

A computing environment with methods for monitoring access to an open network, such as a WAN or the Internet, is described. The system includes one or more clients, each operating applications or processes (e.g., Netscape Navigator™ or Microsoft Internet Explorer™ browser software) requiring Internet (or other open network) access (e.g., an Internet connection to one or more Web servers). Client-based monitoring and filtering of access is provided in conjunction with a centralized enforcement supervisor. The supervisor maintains access rules for the client-based filtering and verifies the existence and proper operation of the client-based filter application. Access rules which can be defined can specify criteria such as total time a user can be connected to the Internet (e.g., per day, week, month, or the like), time a user can interactively use the Internet (e.g., per day, week, month, or the like), a list of applications or application versions that a user can or cannot use in order to access the Internet, a list of URLs (or WAN addresses) that a user application can (or cannot) access, a list of protocols or protocol components (such as Java Script™) that a user application can or cannot use, and rules to determine what events should be logged (including how long are logs to be kept). By intercepting process loading and unloading and keeping a list of currently-active processes, each client process can be checked for various characteristics, including checking executable names, version numbers, executable file checksums, version header details, configuration settings, and the like. With this information, the system can determine if a particular process in question should have access to the Internet and what kind of access (i.e., protocols, Internet addresses, time limitations, and the like) is permissible for the given specific user.

30 Claims, 38 Drawing Sheets



OTHER PUBLICATIONS

Croker, D., "RFC 822—Standard for the format of ARPA Internet Text Messages," Department of Electrical Engineering, University of Delaware, Aug. 13, 1982, pp. 1–47.

Postel, J. and Reynolds, J., "RFC 959—File Transfer Protocol (FTP)," Information Science Institute, University of Southern California, Oct. 1985, pp. 1–47.

Kantor, B. (U.C. San Diego) and Lapsley, P. (U.C. Berkeley), "RFC 977—Network News Transfer Protocol, " Feb. 1986, pp. 1–27.

Berners–Lee, T., "RFC 1630—Universal Resource Identifiers in WWW," Jun. 1994, pp. 28.

Klensin, J., Freed, N., Rose, M., Stefferud, E. and Crocker, D., "RFC 1869—SMTP Service Extensions," Nov. 1995, pp. 1–11.

Kessler, G. and Shepard, S., "RFC 1739—A Primer On Internet And TCP/IP Tools," Hill Associates, Inc., Dec. 1994, pp. 1–46.

Myers, J. (Carnegie Mellon) and Rose, M. (Dover Beach Consulting, Inc.), "RFC 1939—Post Office Protocol—Version 3," May 1996, pp. 1–23.

Freed, N., "RFC 2034—SMTP Service Extension for Returning Enhanced Error Codes," Innosoft, Oct. 1996, pp. 1–6.

Freed, N., Borenstein, N., Moore, K., Klensin, J. and Postel, J., "RFC 2045/2046/2047/2048/2049—Multipurpose Internet Mail Extensions (MIME), Part 1: Format of Internet Message Bodies, Part 2: Media Types, Part 3: Message Header Extensions for Non–ASCII Text, Part 4: Registration Procedures, Part 5: Conformance Criteria and Examples," Nov. 1996, Part 1: pp. 1–31, Part 2: pp. 1–44, Part 3: pp. 1–15, Part 4: pp. 1–21, Part 5: pp. 1–24.

Crispin, M., "RFC 2060—Internet Message Access Protocol—Version 4rev1," University of Washington, Dec. 1996, pp. 1–82.

Palme, J. (Stockholm University) and Hopmann, A. (Microsoft Corporation), "RFC 2110—MIME E–mail Encapsulation of Aggregate Documents, such as HTML (MHTML)," Mar. 1997, pp. 1–19.

Fielding, R. (U.C. Irvine), Gettys, J. (DEC), Mogul, J. (DEC), Frystyk, H. (MIT/LCS) and Berers–Lee, T. (MIT/LCS), "Hypertext Transfer Protocol–HTTP/1.1," Internet Engineering Task Force (IETF)—Internet Draft, Aug. 12, 1996, pp. 1–52.

Marsh, K., "Win32 Hooks," Microsoft Developer Network Technology Group, Jul. 29, 1993 (revised Feb. 1994), pp. 1–14.

Dawson, D., "Firewalls 101—A Introduction to Ascend Secure Access," Ascend Network Secure Business Unit, Sep. 4, 1996, pp. 1–6.

Semeria, C., "Internet Firewalls and Security—A Technology Overview," 3Com Corporation, Sep. 4, 1996, pp. 1–16.

Felten, E., Balfanz, D., Dean, D. and Wallach, D., "Web Spoofing: An Internet Con Game—Technical Report 540–96," Department of Computer Science, Princeton University, 1996, pp. 1–9

Microsoft Corporation, "Microsoft Technical Notes—Browsing and Windows 95 Networking," 1995, pp. 1–38.

Windows Networking Design Team—Microsoft Corporation, "Microsoft TCP/IP VxD Interface Specification," Oct. 24, 1994, pp. 1–23.

TechNet/Corp. Network Systems/Bus. Systems Div.—Microsoft Corporation, "MS Windows NT 3.5/3.51: TCP/IP Implementation Details," May 22, 1996, pp. 1–65.

Shah, R., "Networking in Windows 95—SunWorld Online, " Nov. 1, 1995, pp. 1–6.

Rickard, J., "Internet Architecture," Boardwatch Magazine, 1996, pp. 1–11.

Microsoft Corporation, "Active Directory Design Specification, Version 1.0," Oct. 25, 1996, pp. 1–111.

Semeria, C., "Understanding IP Addressing—Everything You Ever Wanted To Know," NDS Marketing, 3Com Corporation, Apr. 26, 1996, pp. 1–62.

Hall, M. et al, "Windows Sockets 2 Service Provider Interface, Revision 2.2.0," Stardust Technologies, May 10, 1996, pp. 1–200.

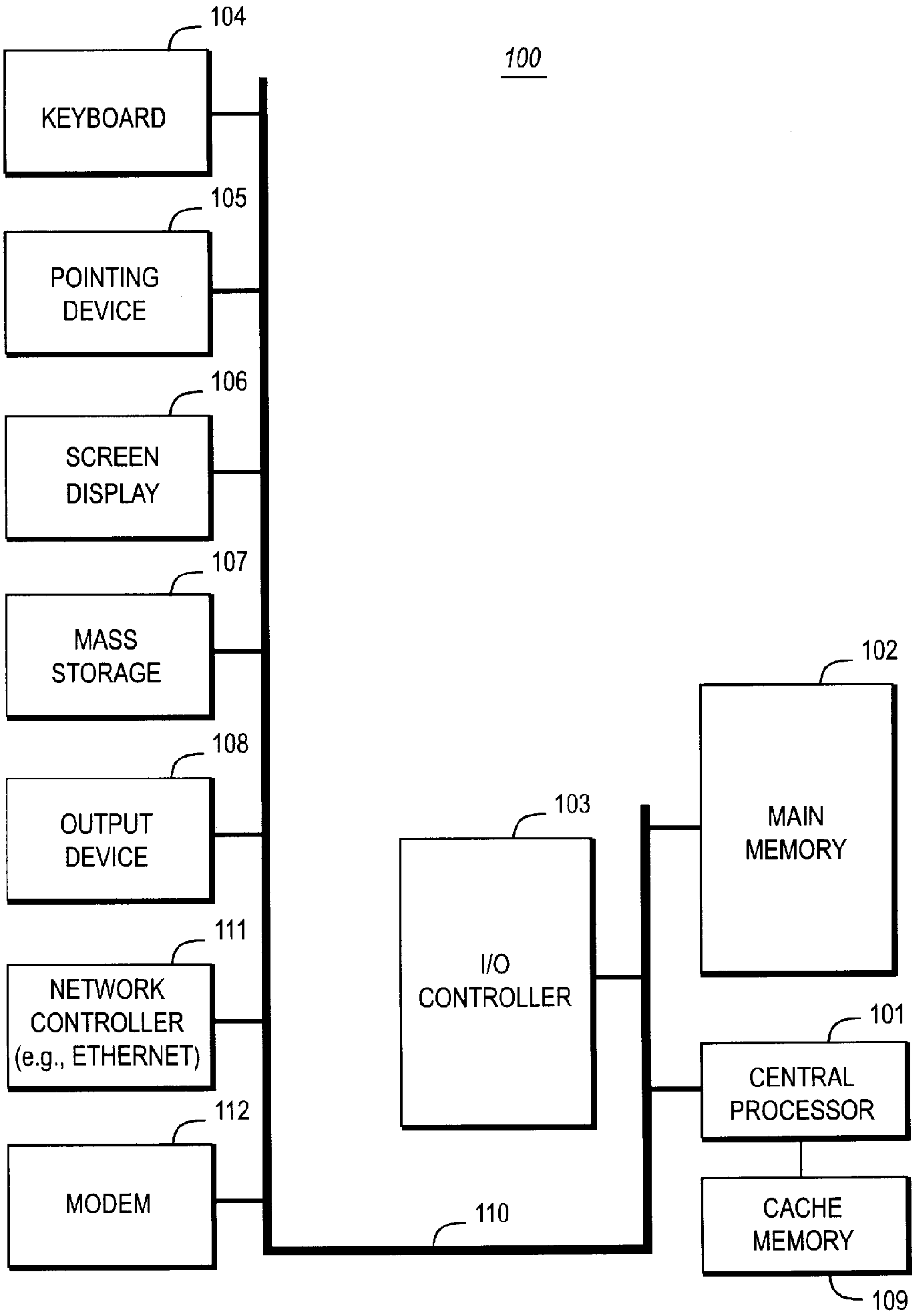

100
104
KEYBOARD
105
POINTING DEVICE
106
SCREEN DISPLAY
107
MASS STORAGE
108
OUTPUT DEVICE
111
NETWORK CONTROLLER (e.g., ETHERNET)
112
MODEM
110
103
I/O CONTROLLER
102
MAIN MEMORY
101
CENTRAL PROCESSOR
CACHE MEMORY
109

FIG. 1

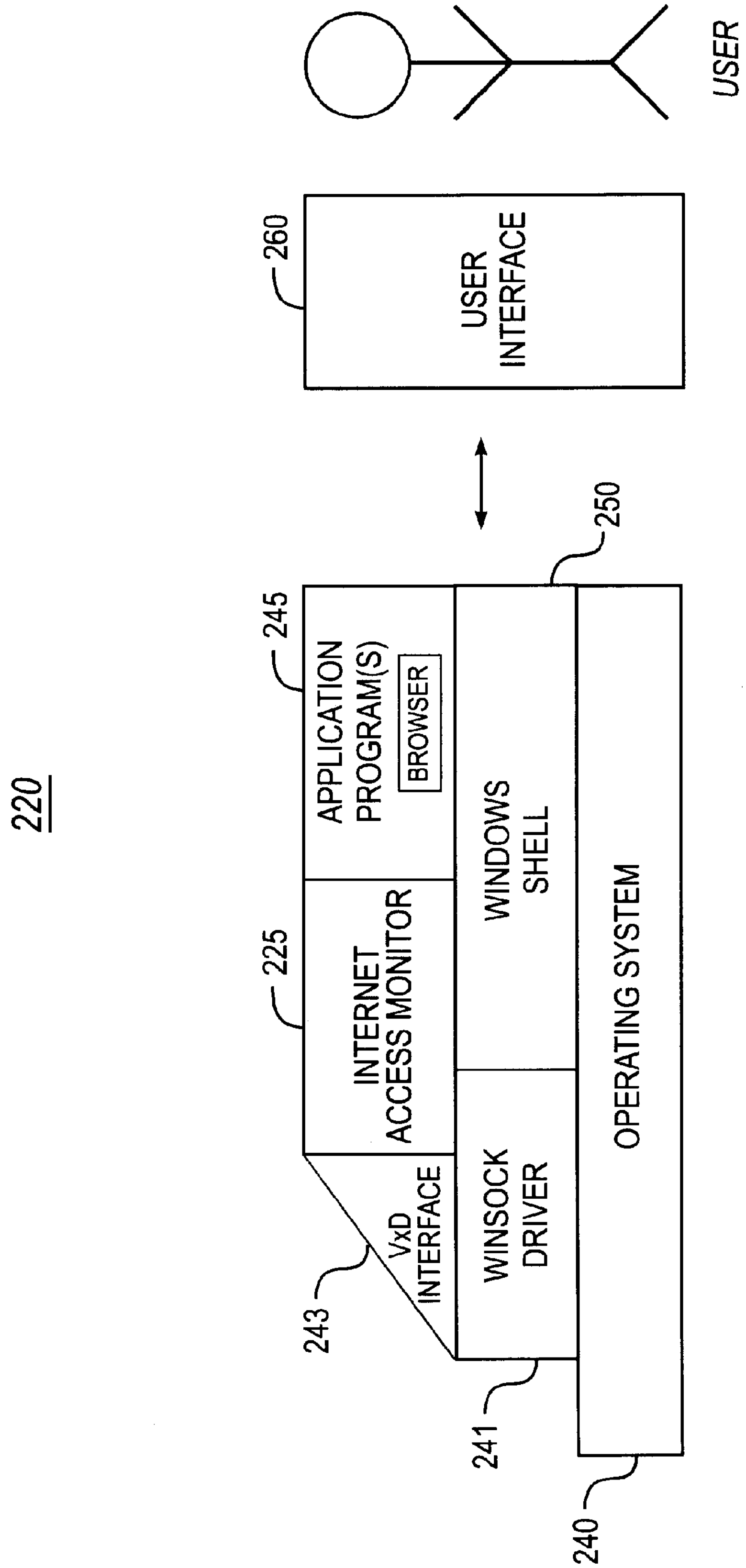

220
USER
260
USER INTERFACE
250
245
APPLICATION PROGRAM(S)
BROWSER
225
INTERNET ACCESS MONITOR
WINDOWS SHELL
243
VxD INTERFACE
WINSOCK DRIVER
241
OPERATING SYSTEM
240

FIG. 2

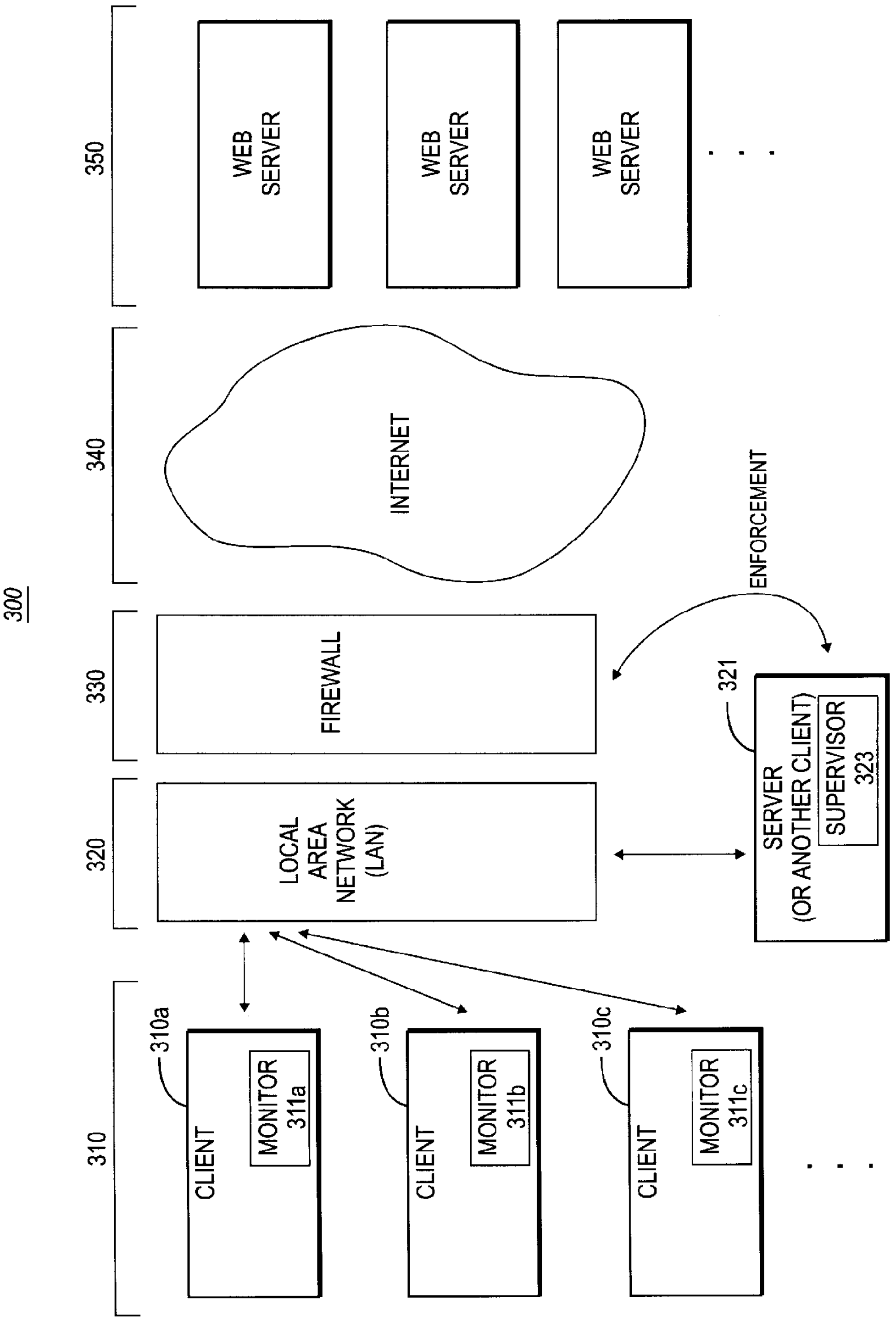

300
310
CLIENT
MONITOR 311a
310a
CLIENT
MONITOR 311b
310b
CLIENT
MONITOR 311c
310c
. . .
320
LOCAL AREA NETWORK (LAN)
SERVER (OR ANOTHER CLIENT)
SUPERVISOR 323
321
330
FIREWALL
ENFORCEMENT
340
INTERNET
350
WEB SERVER
WEB SERVER
WEB SERVER
. . .

FIG. 3A

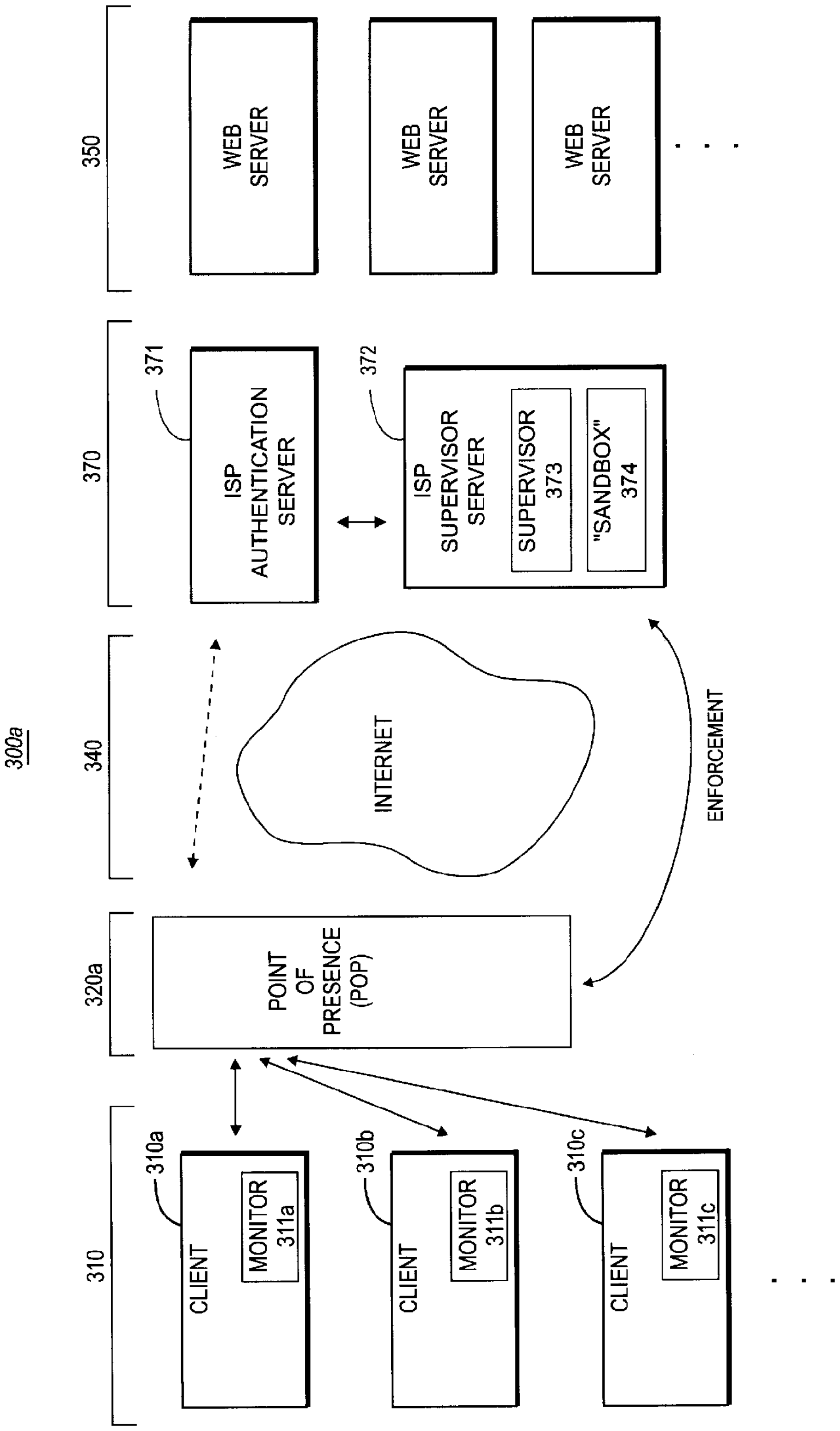

300a
310
310a
CLIENT
MONITOR 311a
310b
CLIENT
MONITOR 311b
310c
CLIENT
MONITOR 311c
320a
POINT OF PRESENCE (POP)
340
INTERNET
ENFORCEMENT
370
371
ISP AUTHENTICATION SERVER
372
ISP SUPERVISOR SERVER
SUPERVISOR 373
"SANDBOX" 374
350
WEB SERVER
WEB SERVER
WEB SERVER

FIG. 3B

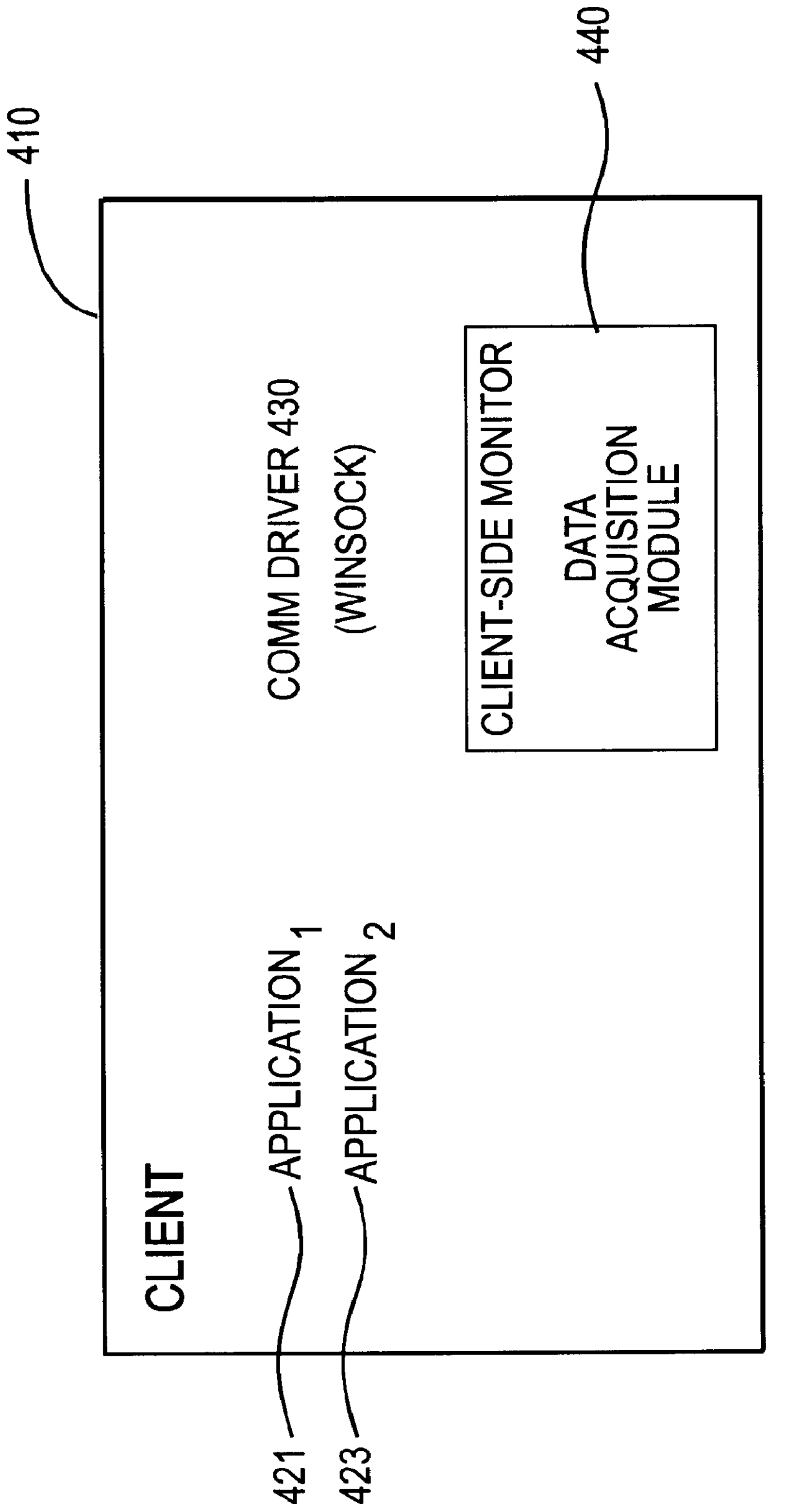

410
440
CLIENT
APPLICATION 1
APPLICATION 2
COMM DRIVER 430
(WINSOCK)
CLIENT-SIDE MONITOR
DATA
ACQUISITION
MODULE
421
423

FIG. 4

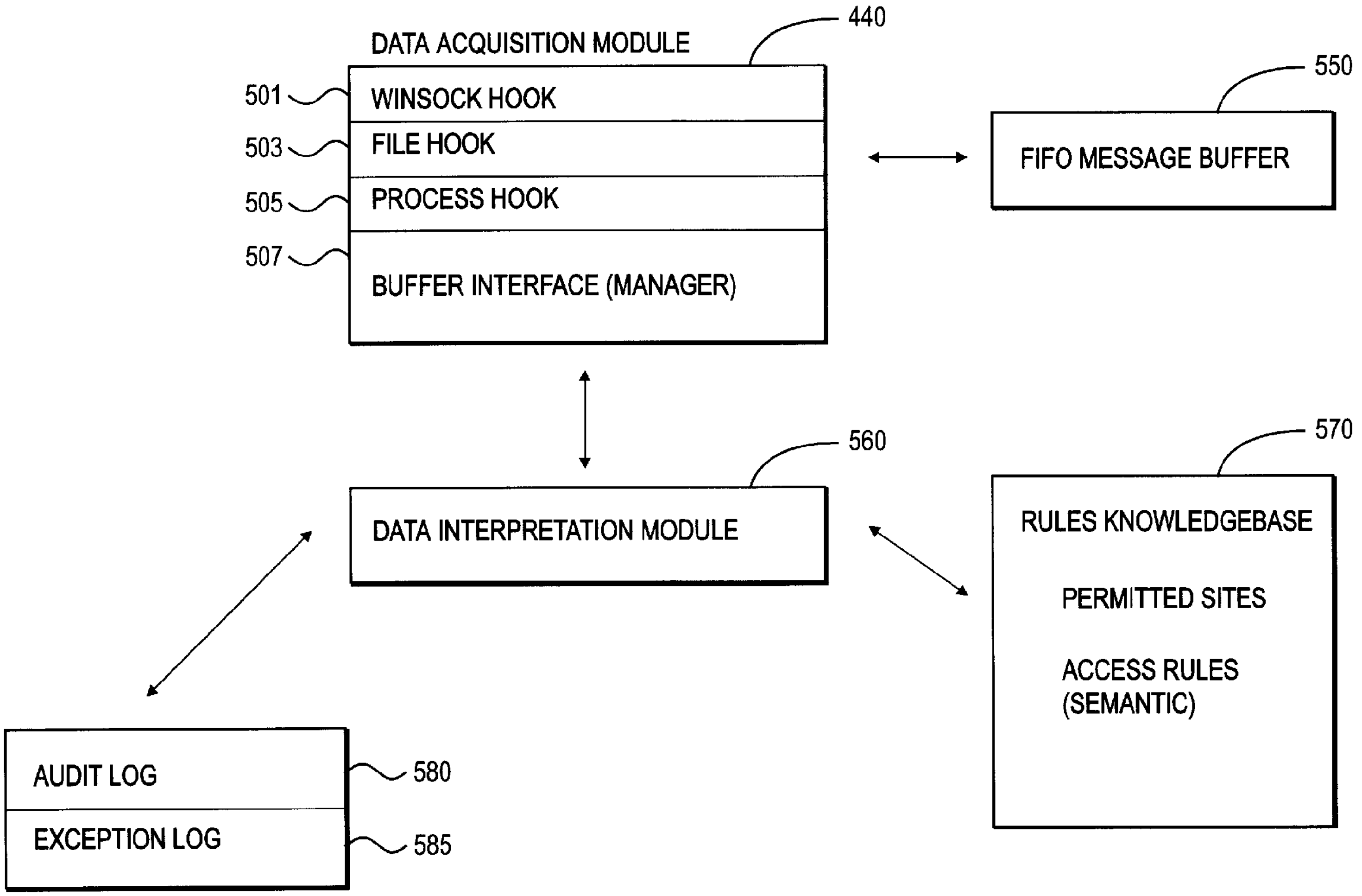
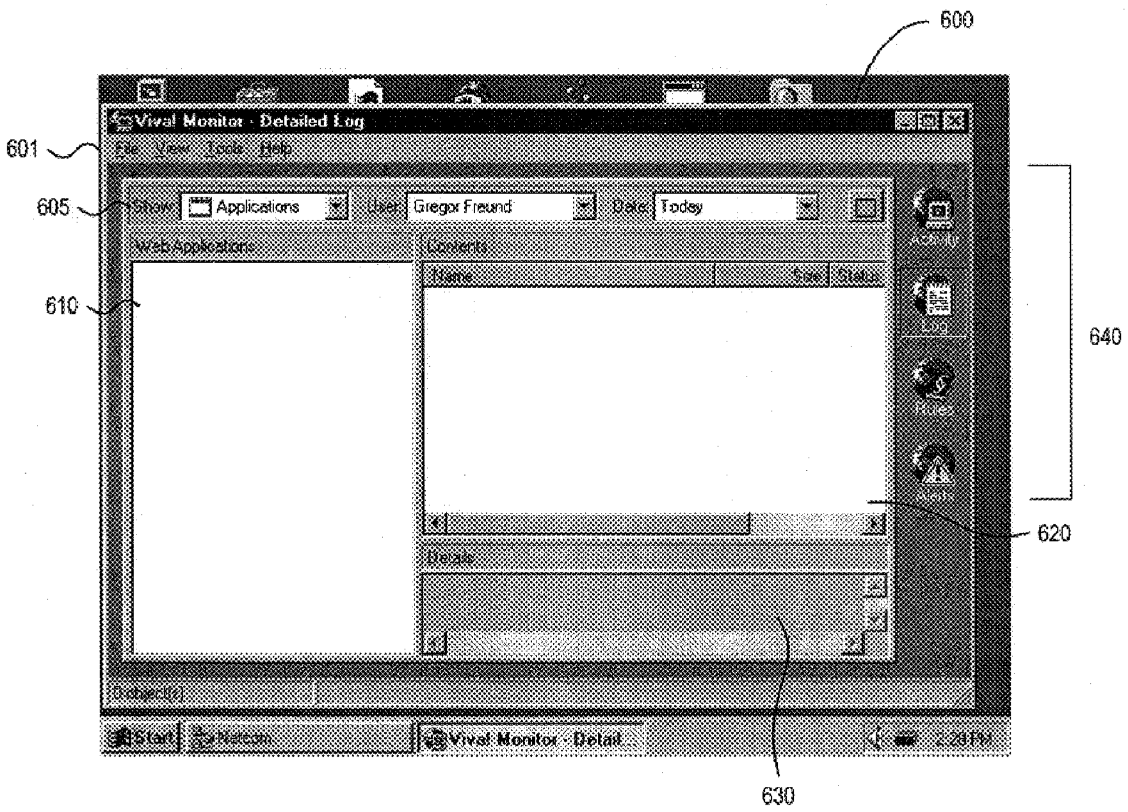

440
DATA ACQUISITION MODULE
501
WINSOCK HOOK
503
FILE HOOK
505
PROCESS HOOK
507
BUFFER INTERFACE (MANAGER)
550
FIFO MESSAGE BUFFER
560
DATA INTERPRETATION MODULE
570
RULES KNOWLEDGEBASE
PERMITTED SITES
ACCESS RULES
(SEMANTIC)
AUDIT LOG
580
EXCEPTION LOG
585

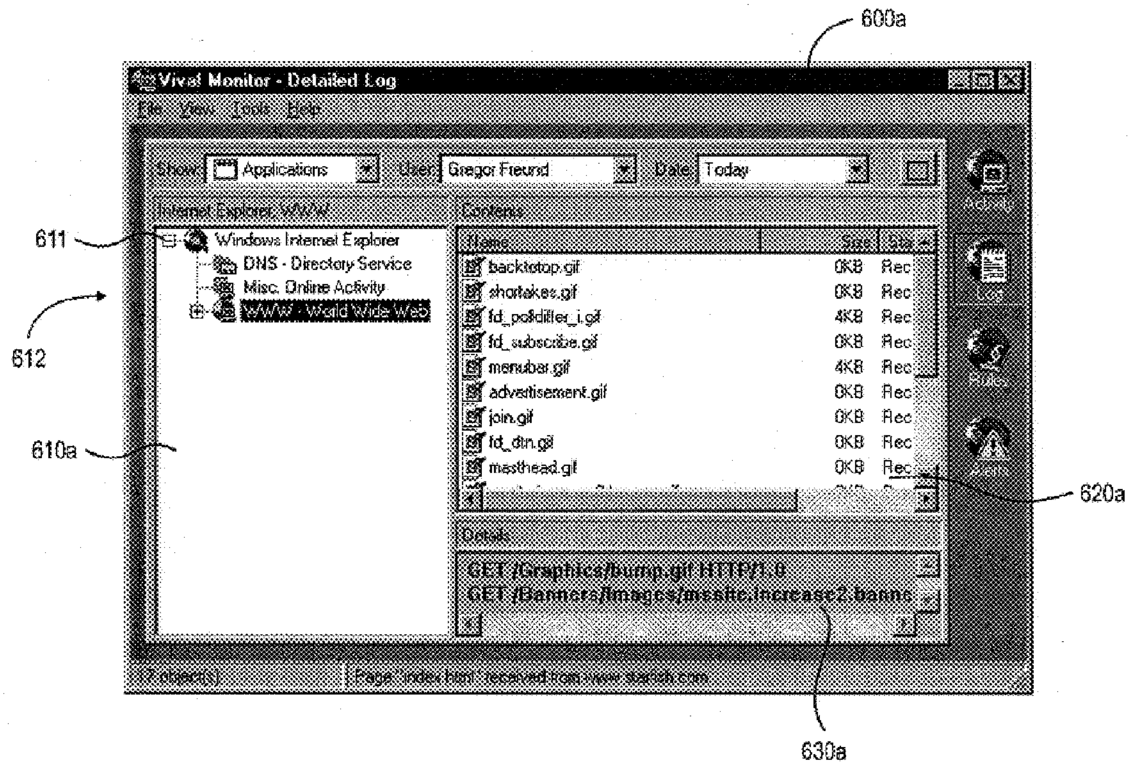
FIG. 5



Vival Monitor - Detailed Log

Show: Applications User: Gregor Freund Date: Today

FIG. 6A



Show: Applications User: Gregor Freund Date: Today

Internet Explorer, WWW

- Windows Internet Explorer
  - DNS - Directory Service
  - Misc. Online Activity
  - WWW - World Wide Web

Contents

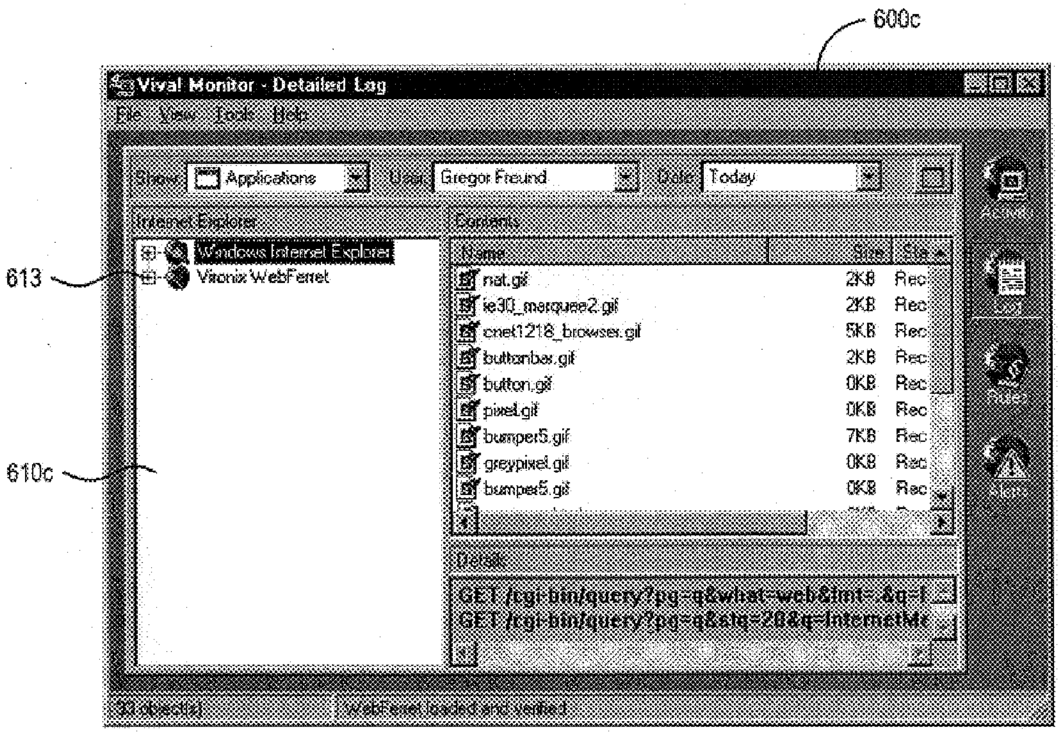
Name	Size	Sta
backtotop.gif	0KB	Rec
shortakes.gif	0KB	Rec
fd_polldiffer_i.gif	4KB	Rec
fd_subscribe.gif	0KB	Rec
menubar.gif	4KB	Rec
advertisement.gif	0KB	Rec
join.gif	0KB	Rec
fd_dtn.gif	0KB	Rec
masthead.gif	0KB	Rec

Details

```
GET /Graphics/bump.gif HTTP/1.0
GET /Banners/Images/mssite.increase2.banne
```

17 object(s) | Page "index.html" received from www.starish.com

FIG. 6B



Show: Applications User: Gregor Freund Date: Today

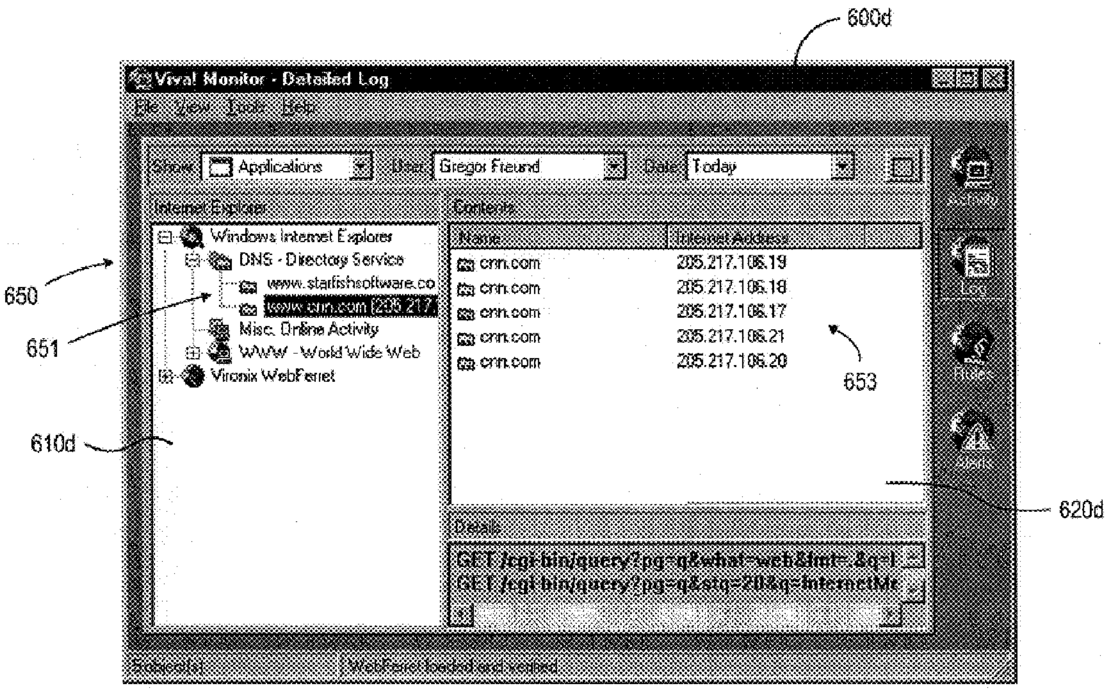
- Windows Internet Explorer
- Vironix WebFerret

Name	Size	Sta
nat.gif	2KB	Rec
ie30_marquee2.gif	2KB	Rec
cnet1218_browser.gif	5KB	Rec
buttonbar.gif	2KB	Rec
button.gif	0KB	Rec
pixel.gif	0KB	Rec
bumper5.gif	7KB	Rec
greypixel.gif	0KB	Rec
bumper5.gif	0KB	Rec

```
GET /cgi-bin/query?pg=q&what=web&fmt=.&q=I
GET /cgi-bin/query?pg=q&stq=20&q=InternetMr
```

33 object(s) WebFerret loaded and verified

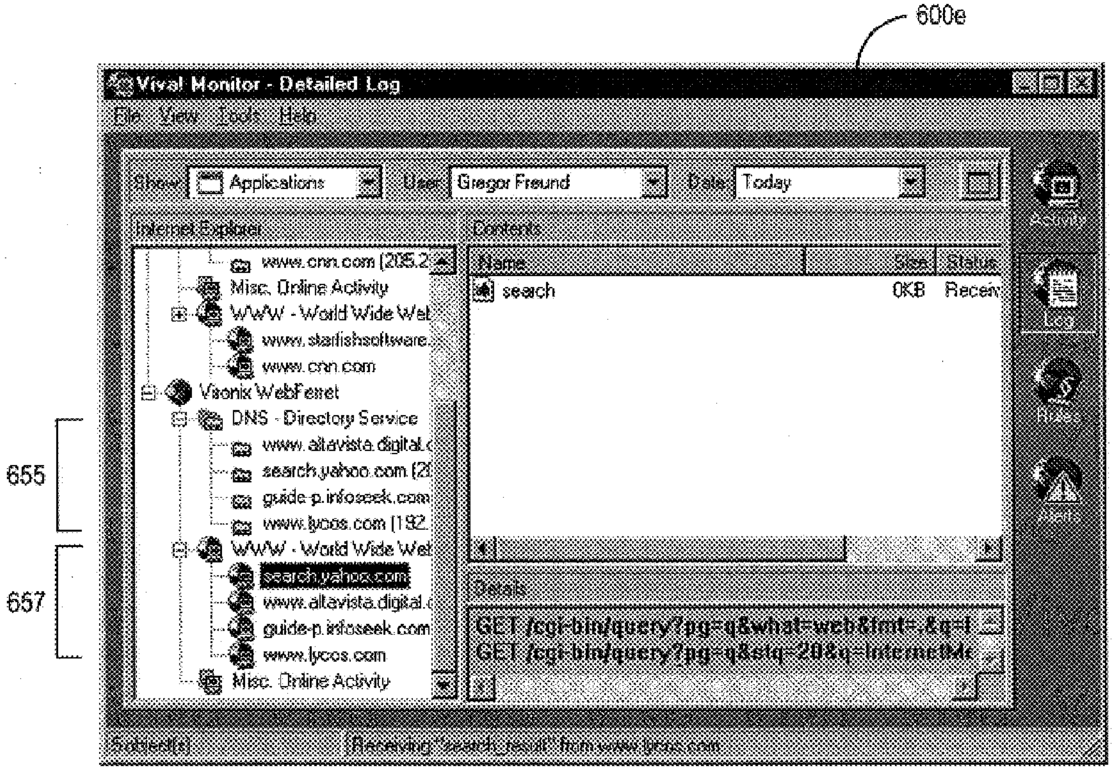
FIG. 6C



Name	Internet Address
cnn.com	205.217.106.19
cnn.com	205.217.106.18
cnn.com	205.217.106.17
cnn.com	205.217.106.21
cnn.com	205.217.106.20

```
GET /cgi-bin/query?pg=q&what=web&fmt=.&q=l
GET /cgi-bin/query?pg=q&stq=20&q=InternetMr
```

FIG. 6D

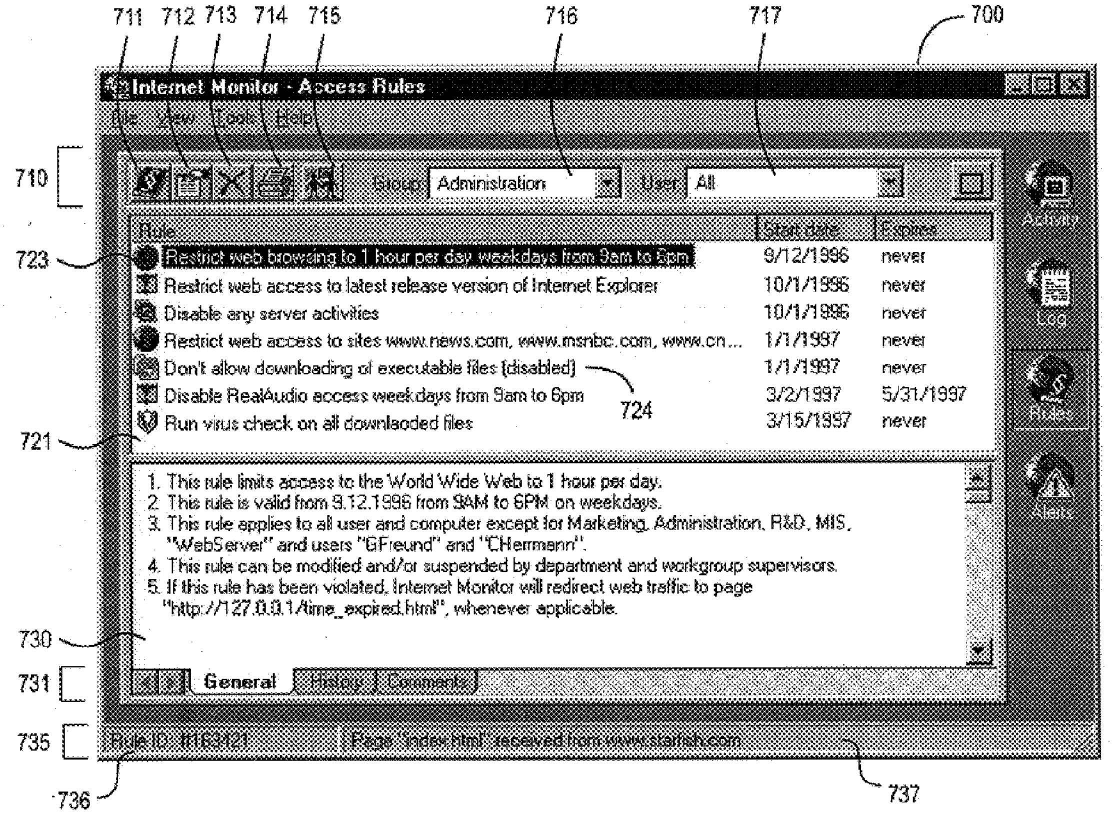


Name	Size	Status
search	0KB	Receiv

```
GET /cgi-bin/query?pg=q&what=web&fmt=.&q=I
GET /cgi-bin/query?pg=q&stq=20&q=InternetMc
```

5 object(s) Receiving "search_result" from www.lycos.com

FIG. 6E



Internet Monitor - Access Rules

Group: Administration

User: All

Rule	Start date	Expires
Restrict web browsing to 1 hour per day weekdays from 9am to 6pm	9/12/1996	never
Restrict web access to latest release version of Internet Explorer	10/1/1996	never
Disable any server activities	10/1/1996	never
Restrict web access to sites www.news.com, www.msnbc.com, www.cn...	1/1/1997	never
Don't allow downloading of executable files [disabled]	1/1/1997	never
Disable RealAudio access weekdays from 9am to 6pm	3/2/1997	5/31/1997
Run virus check on all downloaded files	3/15/1997	never

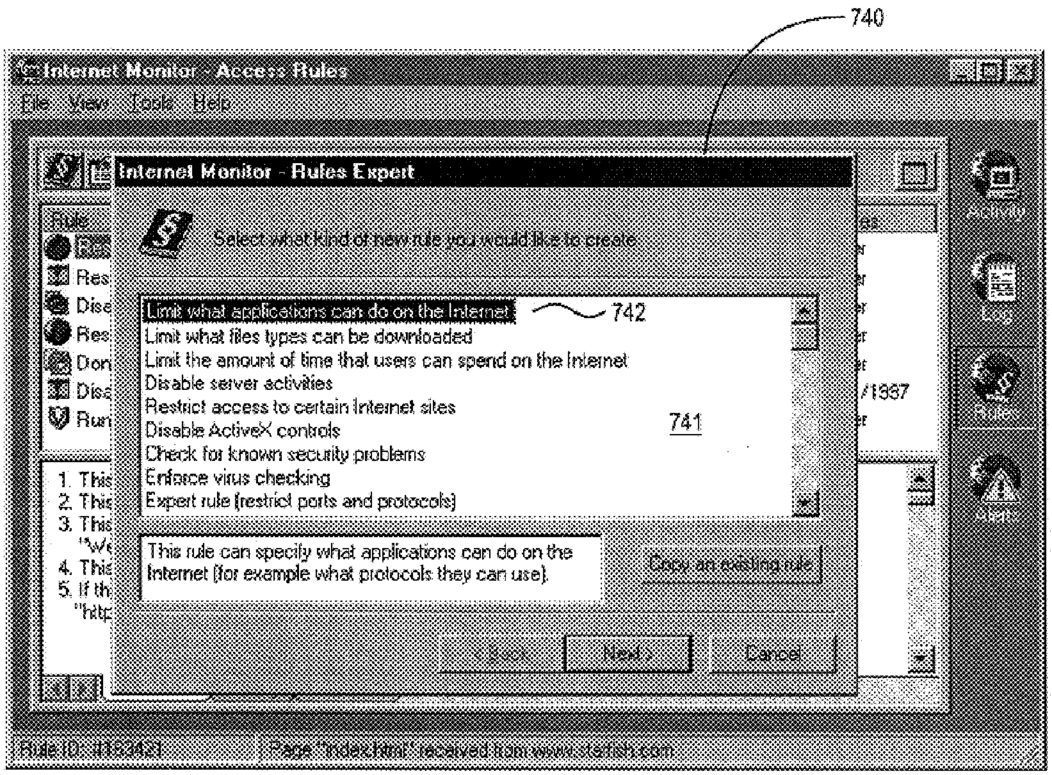
1. This rule limits access to the World Wide Web to 1 hour per day.
2. This rule is valid from 9.12.1996 from 9AM to 6PM on weekdays.
3. This rule applies to all user and computer except for Marketing, Administration, R&D, MIS, "WebServer" and users "GFreund" and "CHermann".
4. This rule can be modified and/or suspended by department and workgroup supervisors.
5. If this rule has been violated, Internet Monitor will redirect web traffic to page "http://127.0.0.1/time_expired.html", whenever applicable.

General | History | Comments

Rule ID: #163421

Page "index.html" received from www.starfish.com

FIG. 7A



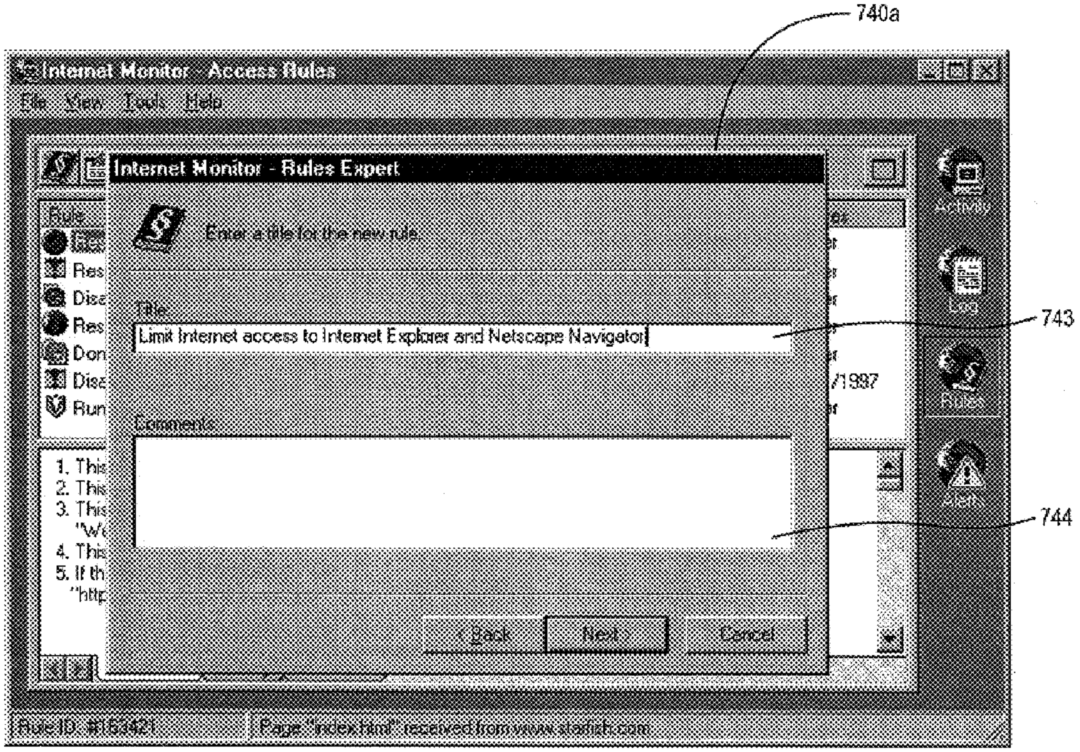
Select what kind of new rule you would like to create.

- Limit what applications can do on the Internet
- Limit what files types can be downloaded
- Limit the amount of time that users can spend on the Internet
- Disable server activities
- Restrict access to certain Internet sites
- Disable ActiveX controls
- Check for known security problems
- Enforce virus checking
- Expert rule (restrict ports and protocols)

This rule can specify what applications can do on the Internet (for example what protocols they can use).

Rule ID: #183421 Page "index.html" received from www.starfish.com

FIG. 7B



Enter a title for the new rule.

Title

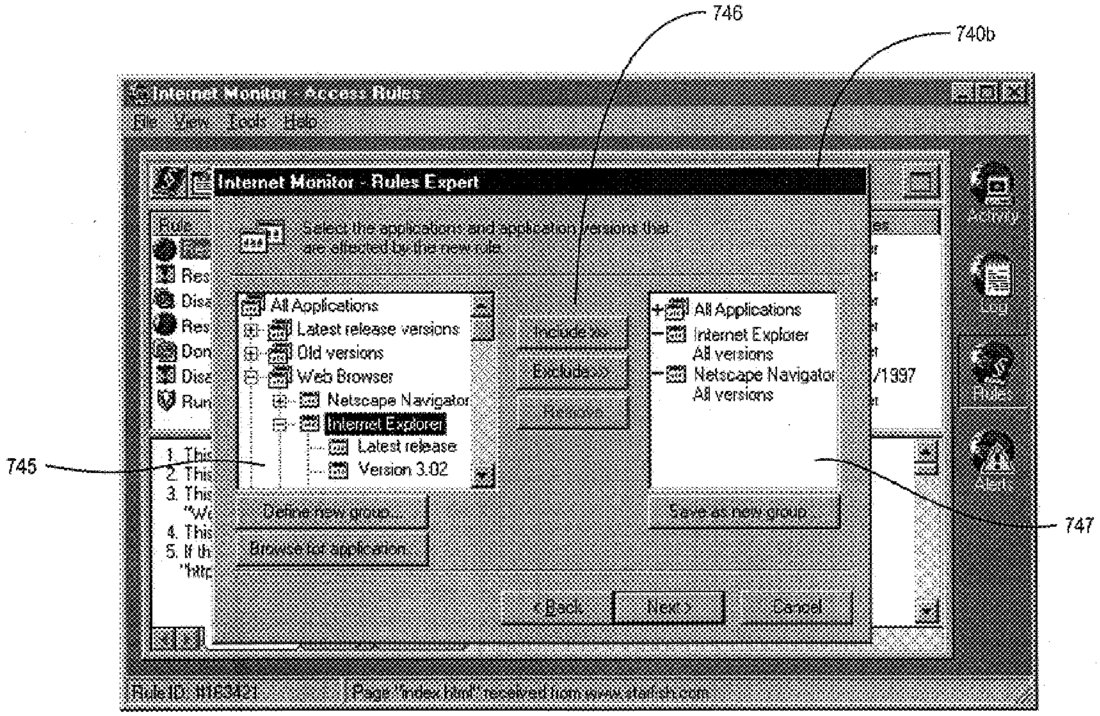
Limit Internet access to Internet Explorer and Netscape Navigator

Comments

Rule ID: #153421

Page "index.html" received from www.starfish.com

FIG. 7C

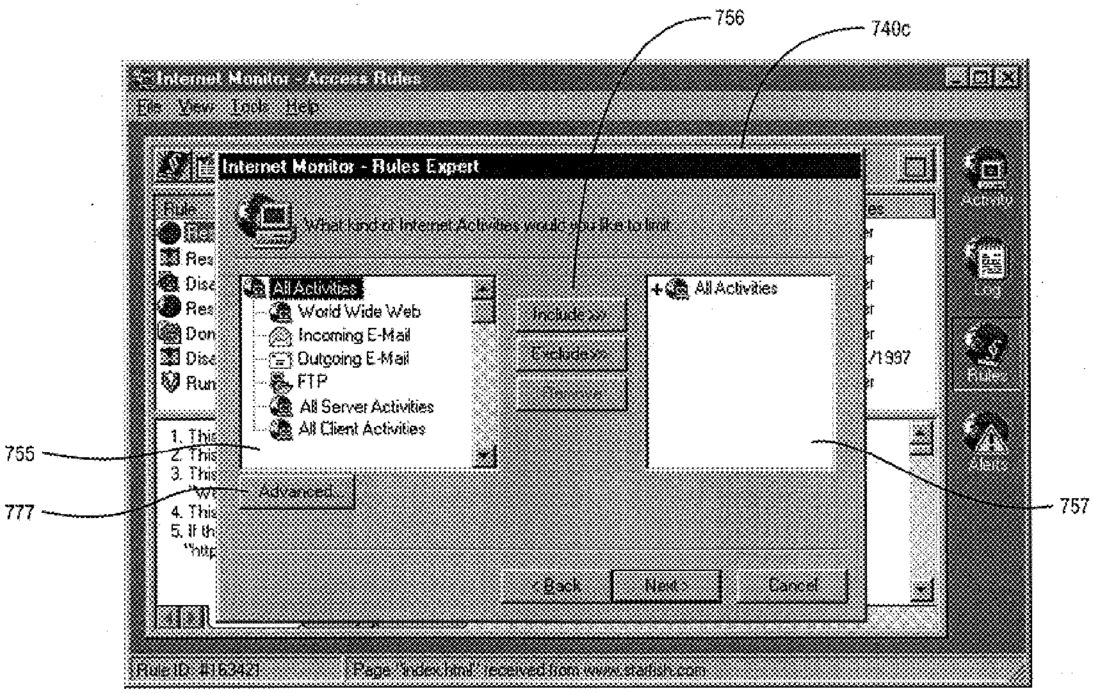


Select the applications and application versions that are affected by the new rule.

Rule ID: #1E3421

Page "index.html" received from www.starfish.com

FIG. 7D



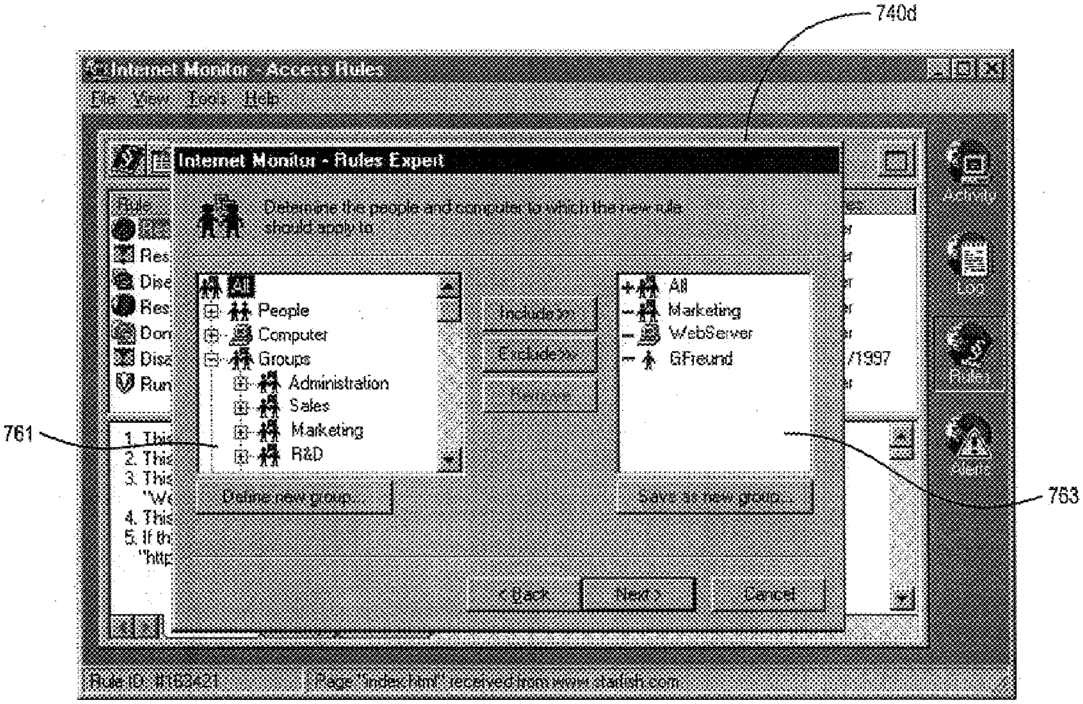
What kind of Internet Activities would you like to limit

1. This
2. This
3. This
"W
4. This
5. If th
"http

Rule ID: #163421

Page "index.html" received from www.starfish.com

FIG. 7E



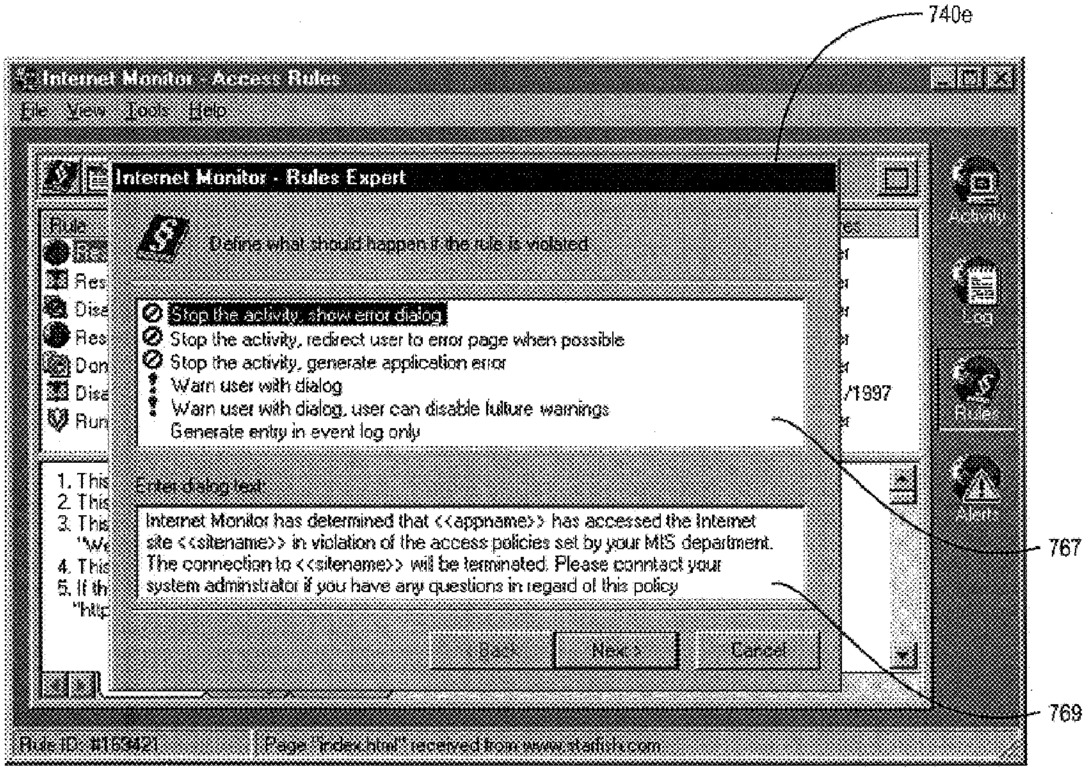
Internet Monitor - Access Rules

Internet Monitor - Rules Expert

Determine the people and computer to which the new rule should apply to

Rule ID: #163421 Page "index.html" received from www.starfish.com

FIG. 7F



Define what should happen if the rule is violated

- Stop the activity, show error dialog
- Stop the activity, redirect user to error page when possible
- Stop the activity, generate application error
- Warn user with dialog
- Warn user with dialog, user can disable fulture warnings
- Generate entry in event log only

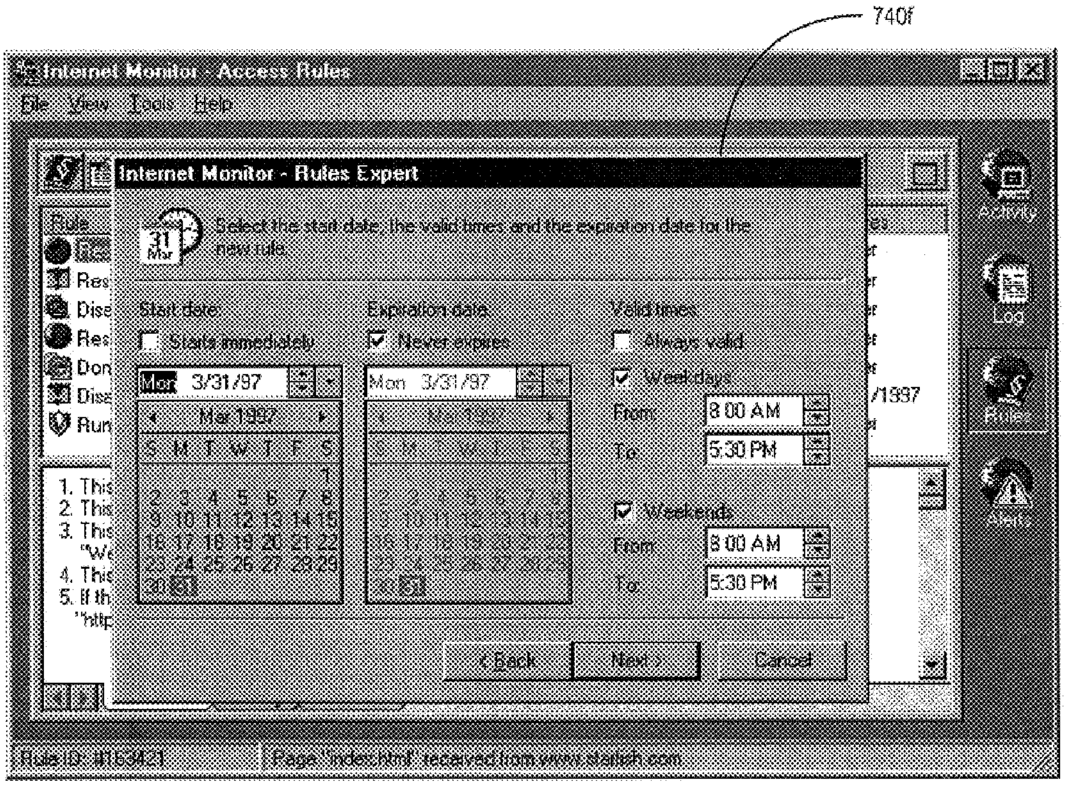
Enter dialog text:

Internet Monitor has determined that <<appname>> has accessed the Internet site <<sitename>> in violation of the access policies set by your MIS department. The connection to <<sitename>> will be terminated. Please conntact your system adminstrator if you have any questions in regard of this policy

< Back | Next > | Cancel

Rule ID: #163421 | Page "index.html" received from www.starfish.com

FIG. 7G



Select the start date, the valid times and the expiration date for the new rule.

Start date:
□ Starts immediately
Mon 3/31/97
Mar 1997

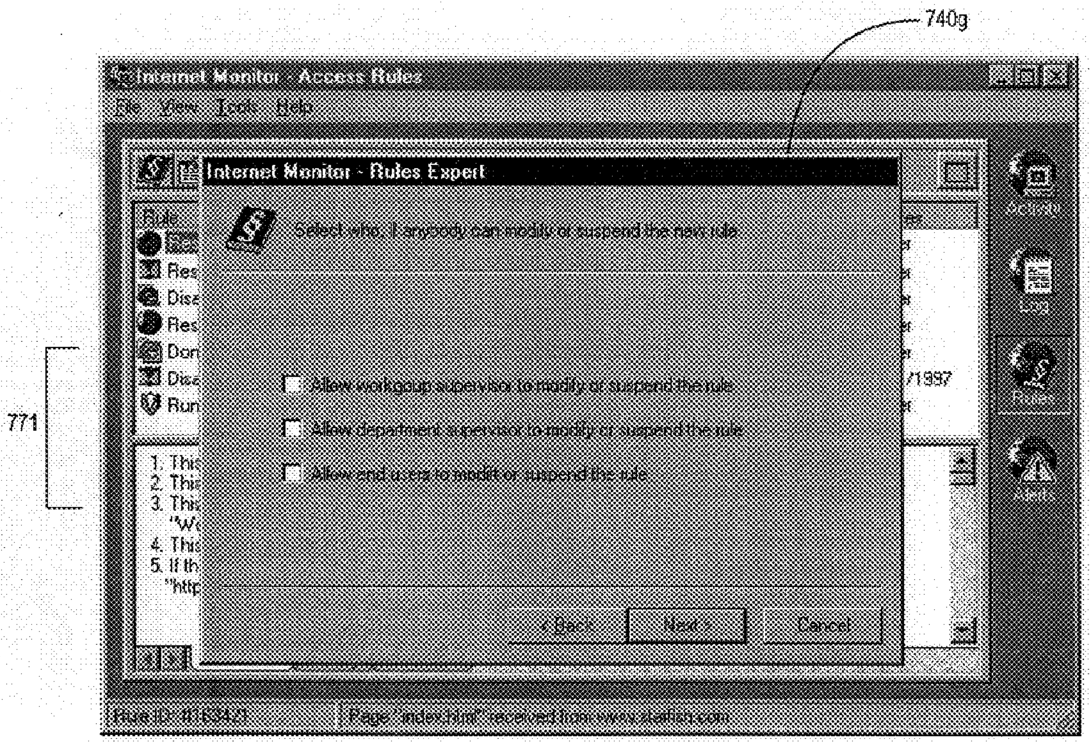
S	M	T	W	T	F	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Expiration date:
☑ Never expires
Mon 3/31/97

Valid times:
□ Always valid
☑ Weekdays
From: 8:00 AM
To: 5:30 PM
☑ Weekends
From: 8:00 AM
To: 5:30 PM

Rule ID: #153421
Page "index.html" received from www.starfish.com

FIG. 7H



Select who, if anybody can modify or suspend the new rule

□ Allow workgoup supervisor to modify or suspend the rule

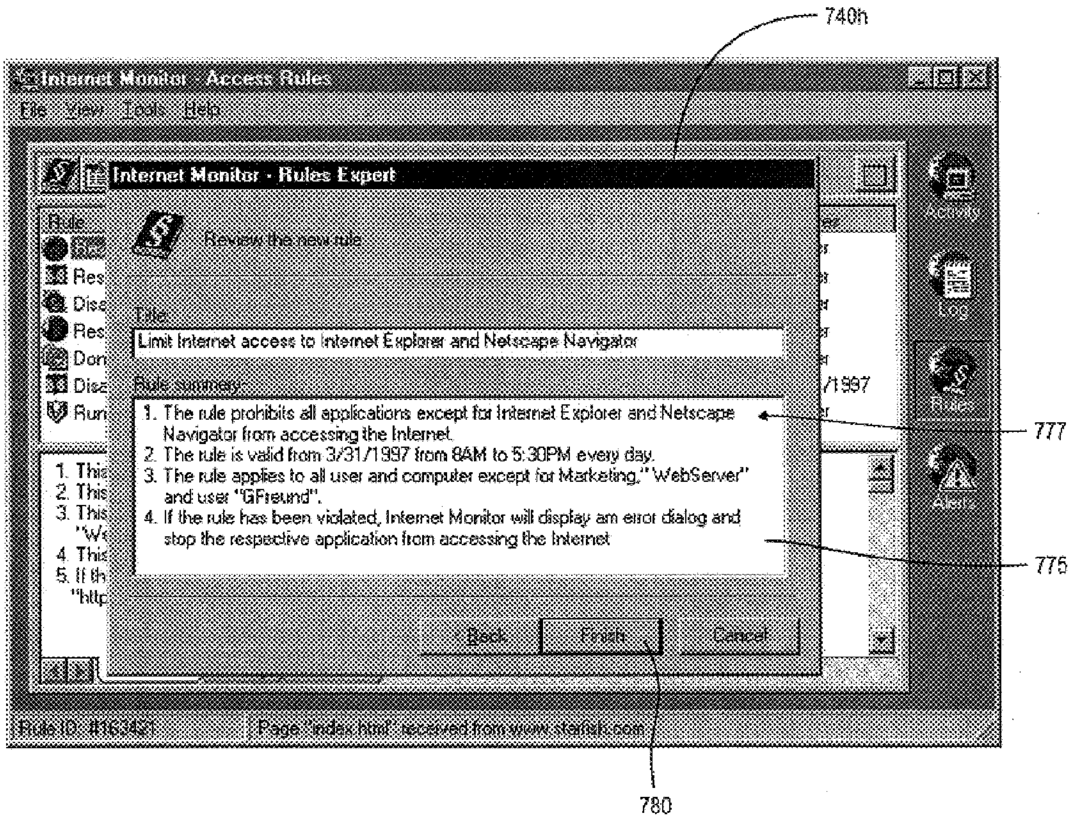
□ Allow department supervisor to modify or suspend the rule

□ Allow end users to modify or suspend the rule

Rule ID: #16342I

Page "index.html" received from www.starfish.com

FIG. 7I



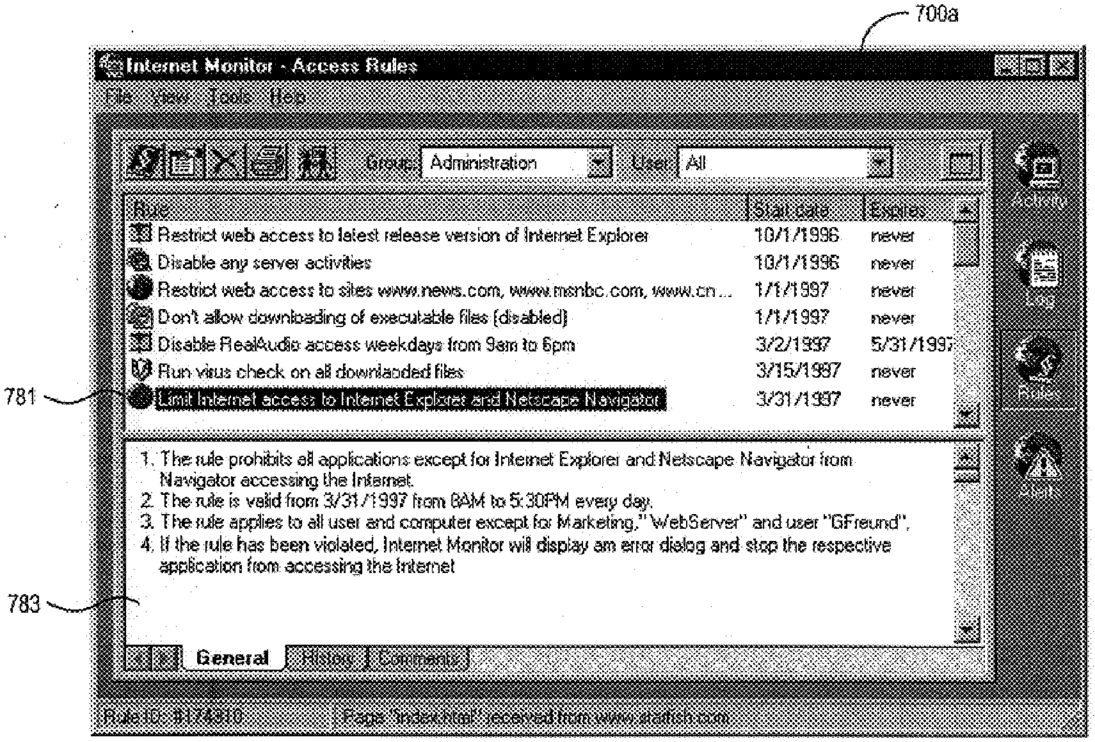
Limit Internet access to Internet Explorer and Netscape Navigator

1. The rule prohibits all applications except for Internet Explorer and Netscape Navigator from accessing the Internet.
2. The rule is valid from 3/31/1997 from 8AM to 5:30PM every day.
3. The rule applies to all user and computer except for Marketing," WebServer" and user "GFreund".
4. If the rule has been violated, Internet Monitor will display am error dialog and stop the respective application from accessing the Internet

Rule ID: #163421

Page "index.html" received from www.starfish.com

FIG. 7J



Internet Monitor - Access Rules

Group: Administration User: All

Rule	Start date	Expires
Restrict web access to latest release version of Internet Explorer	10/1/1996	never
Disable any server activities	10/1/1996	never
Restrict web access to sites www.news.com, www.msnbc.com, www.cn...	1/1/1997	never
Don't allow downloading of executable files (disabled)	1/1/1997	never
Disable RealAudio access weekdays from 9am to 6pm	3/2/1997	5/31/1997
Run virus check on all downloaded files	3/15/1997	never
Limit Internet access to Internet Explorer and Netscape Navigator	3/31/1997	never

1. The rule prohibits all applications except for Internet Explorer and Netscape Navigator from Navigator accessing the Internet.
2. The rule is valid from 3/31/1997 from 8AM to 5:30PM every day.
3. The rule applies to all user and computer except for Marketing," WebServer" and user "GFreund".
4. If the rule has been violated, Internet Monitor will display am error dialog and stop the respective application from accessing the Internet

Rule ID: #174810 Page 'index.html' received from www.starfish.com

FIG. 7K

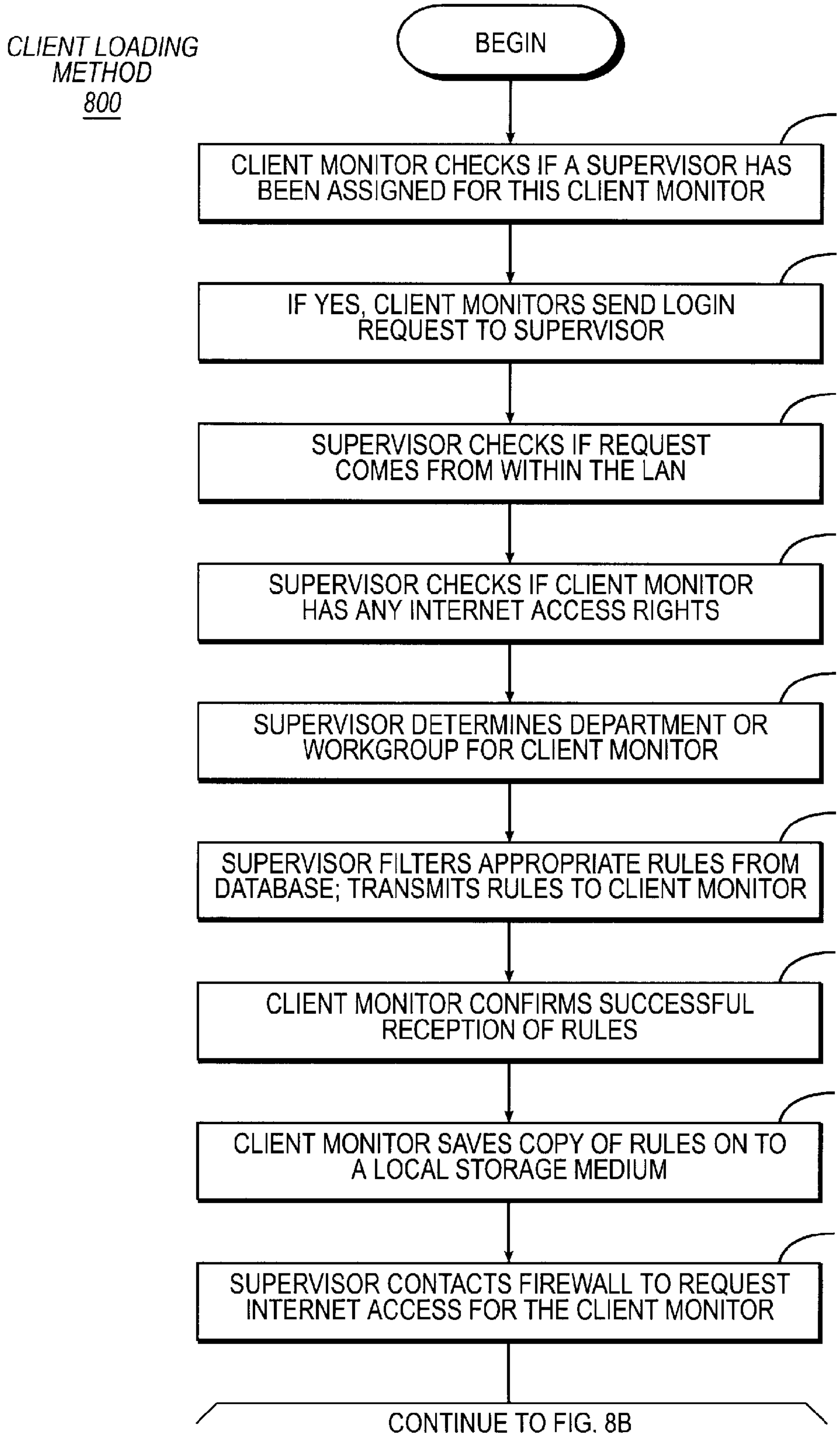

CLIENT LOADING METHOD 800
BEGIN
801
CLIENT MONITOR CHECKS IF A SUPERVISOR HAS BEEN ASSIGNED FOR THIS CLIENT MONITOR
802
IF YES, CLIENT MONITORS SEND LOGIN REQUEST TO SUPERVISOR
803
SUPERVISOR CHECKS IF REQUEST COMES FROM WITHIN THE LAN
804
SUPERVISOR CHECKS IF CLIENT MONITOR HAS ANY INTERNET ACCESS RIGHTS
805
SUPERVISOR DETERMINES DEPARTMENT OR WORKGROUP FOR CLIENT MONITOR
806
SUPERVISOR FILTERS APPROPRIATE RULES FROM DATABASE; TRANSMITS RULES TO CLIENT MONITOR
807
CLIENT MONITOR CONFIRMS SUCCESSFUL RECEPTION OF RULES
808
CLIENT MONITOR SAVES COPY OF RULES ON TO A LOCAL STORAGE MEDIUM
809
SUPERVISOR CONTACTS FIREWALL TO REQUEST INTERNET ACCESS FOR THE CLIENT MONITOR
CONTINUE TO FIG. 8B

FIG. 8A

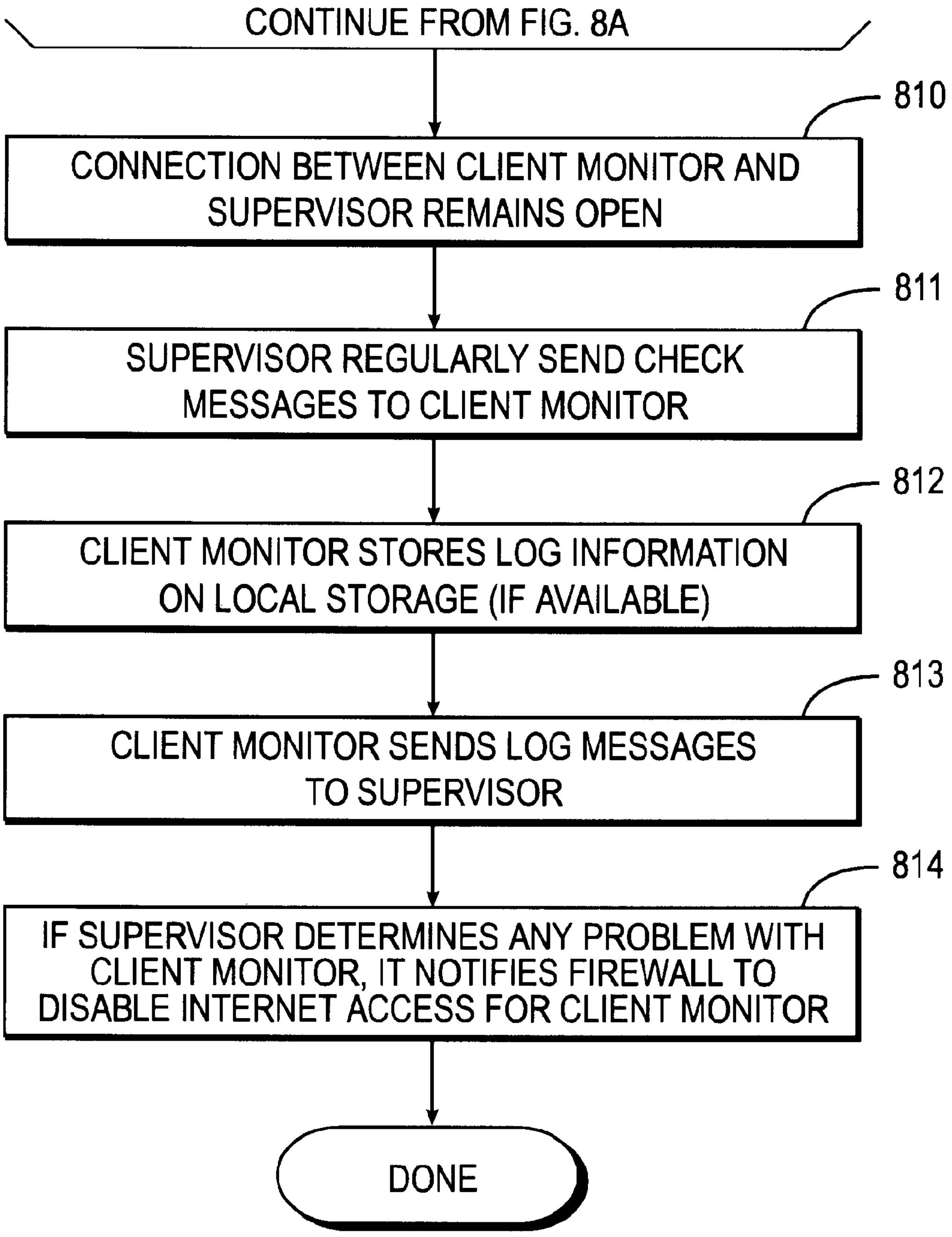

CONTINUE FROM FIG. 8A
810
CONNECTION BETWEEN CLIENT MONITOR AND SUPERVISOR REMAINS OPEN
811
SUPERVISOR REGULARLY SEND CHECK MESSAGES TO CLIENT MONITOR
812
CLIENT MONITOR STORES LOG INFORMATION ON LOCAL STORAGE (IF AVAILABLE)
813
CLIENT MONITOR SENDS LOG MESSAGES TO SUPERVISOR
814
IF SUPERVISOR DETERMINES ANY PROBLEM WITH CLIENT MONITOR, IT NOTIFIES FIREWALL TO DISABLE INTERNET ACCESS FOR CLIENT MONITOR
DONE

FIG. 8B

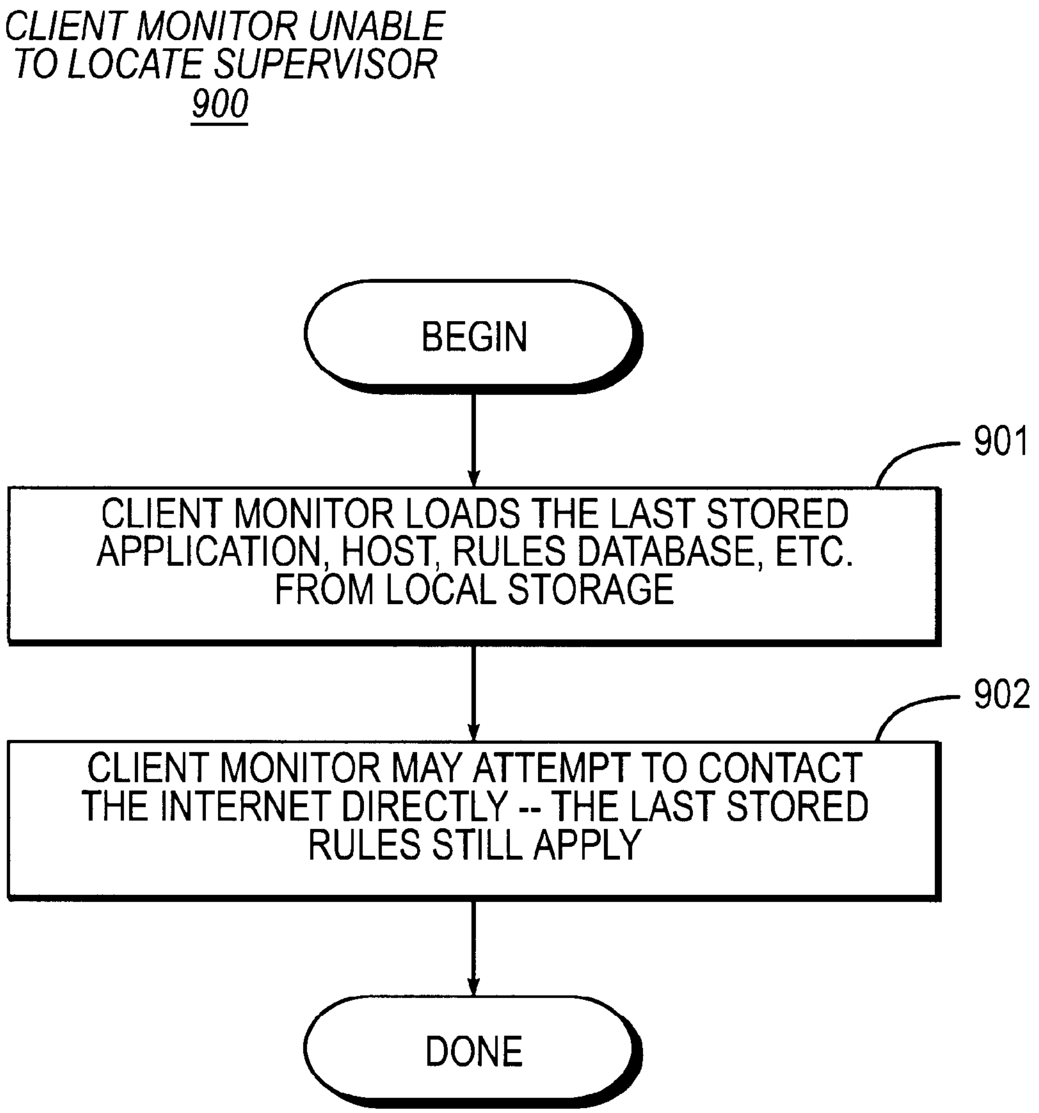

CLIENT MONITOR UNABLE TO LOCATE SUPERVISOR
900
BEGIN
901
CLIENT MONITOR LOADS THE LAST STORED APPLICATION, HOST, RULES DATABASE, ETC. FROM LOCAL STORAGE
902
CLIENT MONITOR MAY ATTEMPT TO CONTACT THE INTERNET DIRECTLY -- THE LAST STORED RULES STILL APPLY
DONE

FIG. 9

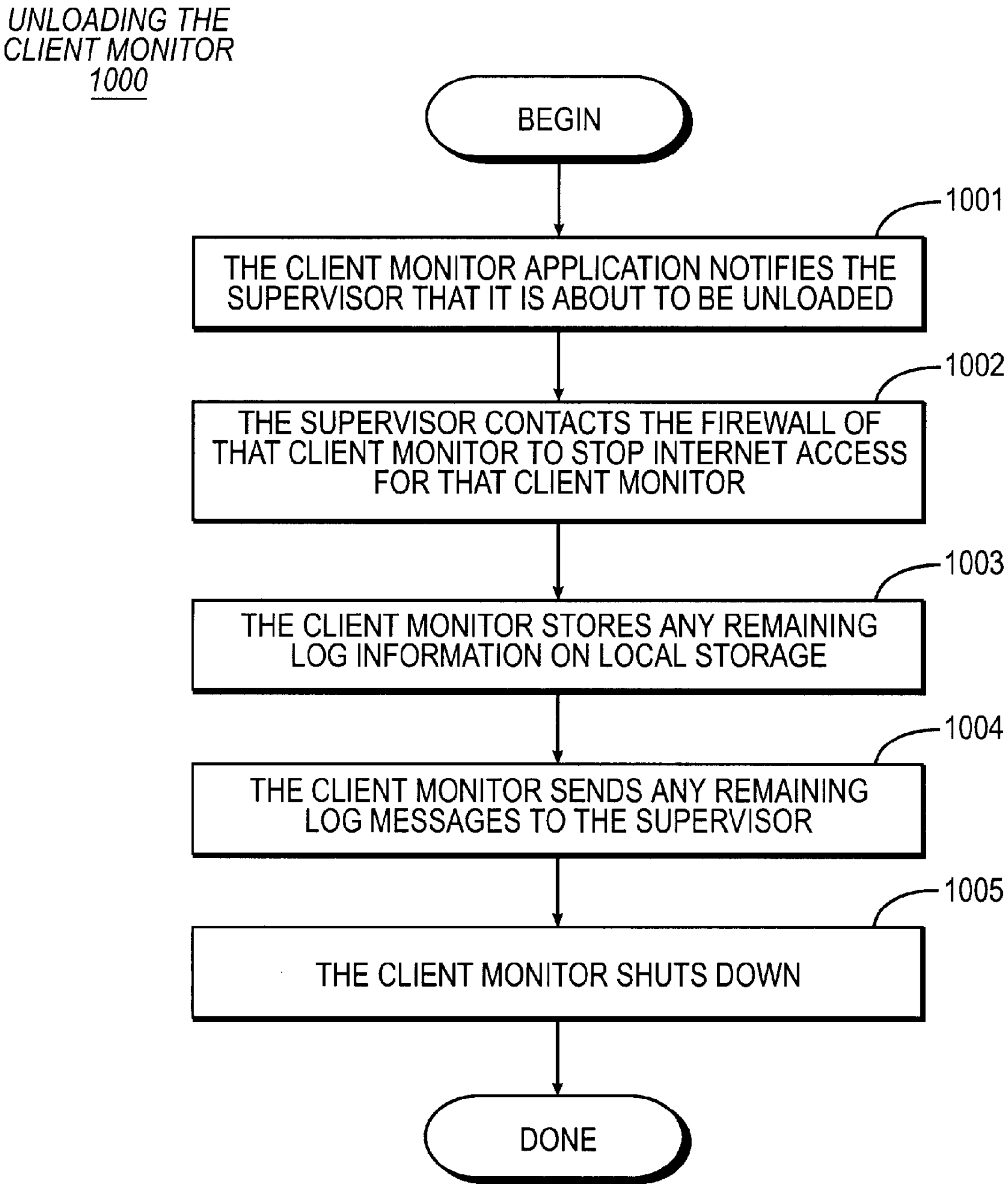

UNLOADING THE CLIENT MONITOR
1000
BEGIN
1001
THE CLIENT MONITOR APPLICATION NOTIFIES THE SUPERVISOR THAT IT IS ABOUT TO BE UNLOADED
1002
THE SUPERVISOR CONTACTS THE FIREWALL OF THAT CLIENT MONITOR TO STOP INTERNET ACCESS FOR THAT CLIENT MONITOR
1003
THE CLIENT MONITOR STORES ANY REMAINING LOG INFORMATION ON LOCAL STORAGE
1004
THE CLIENT MONITOR SENDS ANY REMAINING LOG MESSAGES TO THE SUPERVISOR
1005
THE CLIENT MONITOR SHUTS DOWN
DONE

FIG. 10

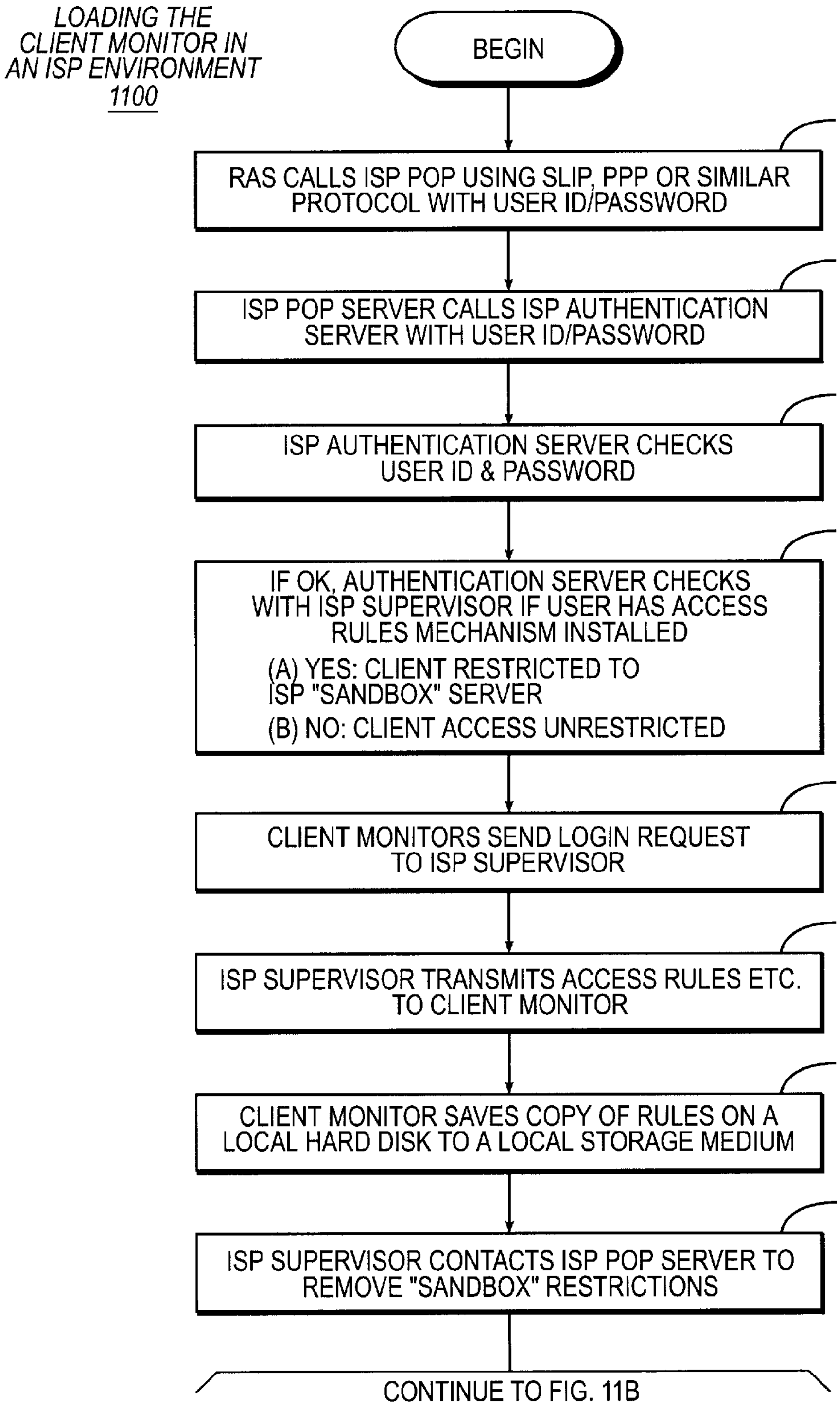

LOADING THE
CLIENT MONITOR IN
AN ISP ENVIRONMENT
1100
BEGIN
1101
RAS CALLS ISP POP USING SLIP, PPP OR SIMILAR PROTOCOL WITH USER ID/PASSWORD
1102
ISP POP SERVER CALLS ISP AUTHENTICATION SERVER WITH USER ID/PASSWORD
1103
ISP AUTHENTICATION SERVER CHECKS USER ID & PASSWORD
1104
IF OK, AUTHENTICATION SERVER CHECKS WITH ISP SUPERVISOR IF USER HAS ACCESS RULES MECHANISM INSTALLED
(A) YES: CLIENT RESTRICTED TO ISP "SANDBOX" SERVER
(B) NO: CLIENT ACCESS UNRESTRICTED
1105
CLIENT MONITORS SEND LOGIN REQUEST TO ISP SUPERVISOR
1106
ISP SUPERVISOR TRANSMITS ACCESS RULES ETC. TO CLIENT MONITOR
1107
CLIENT MONITOR SAVES COPY OF RULES ON A LOCAL HARD DISK TO A LOCAL STORAGE MEDIUM
1108
ISP SUPERVISOR CONTACTS ISP POP SERVER TO REMOVE "SANDBOX" RESTRICTIONS
CONTINUE TO FIG. 11B

FIG. 11A

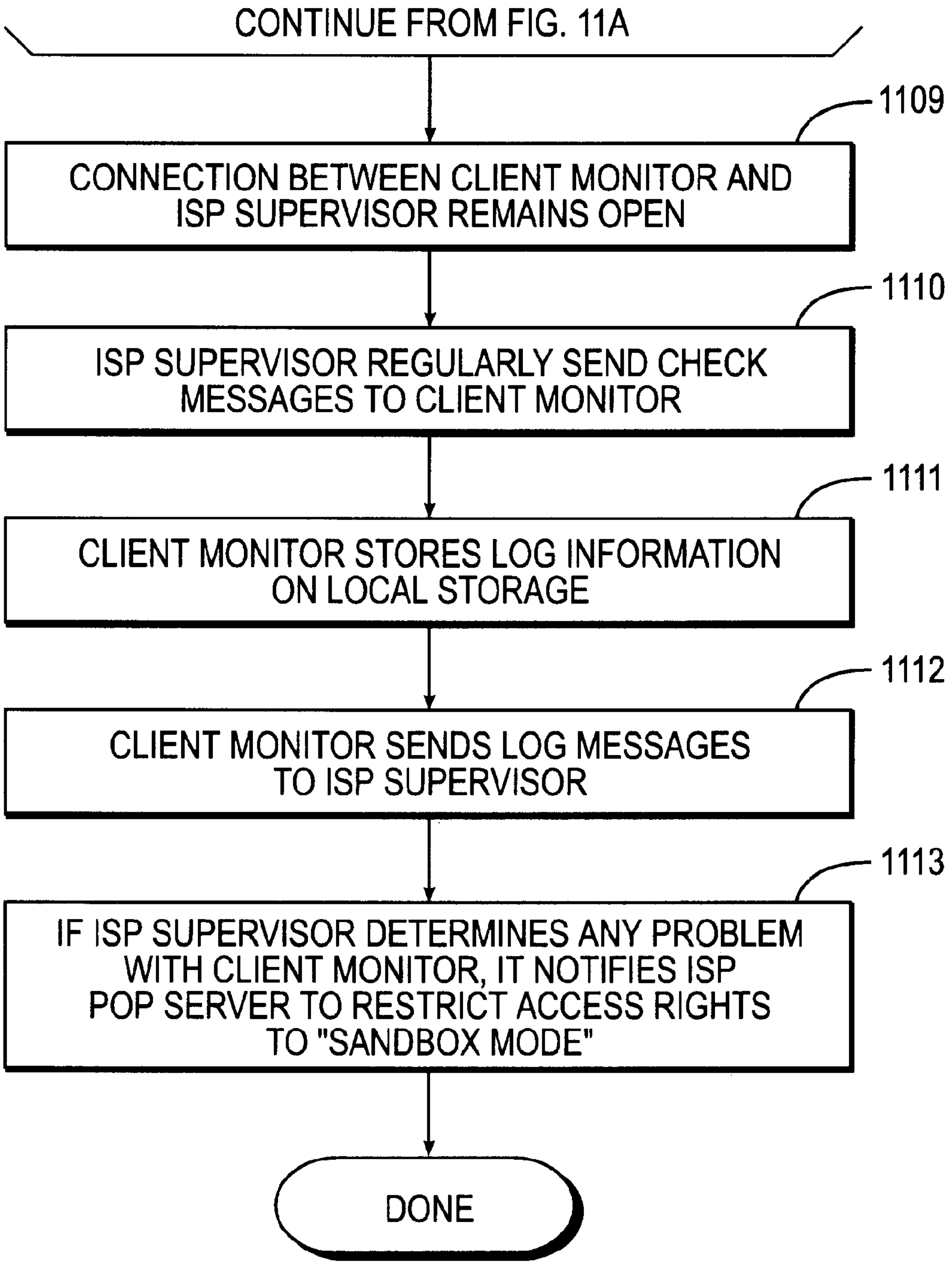

CONTINUE FROM FIG. 11A
1109
CONNECTION BETWEEN CLIENT MONITOR AND ISP SUPERVISOR REMAINS OPEN
1110
ISP SUPERVISOR REGULARLY SEND CHECK MESSAGES TO CLIENT MONITOR
1111
CLIENT MONITOR STORES LOG INFORMATION ON LOCAL STORAGE
1112
CLIENT MONITOR SENDS LOG MESSAGES TO ISP SUPERVISOR
1113
IF ISP SUPERVISOR DETERMINES ANY PROBLEM WITH CLIENT MONITOR, IT NOTIFIES ISP POP SERVER TO RESTRICT ACCESS RIGHTS TO "SANDBOX MODE"
DONE

FIG. 11B

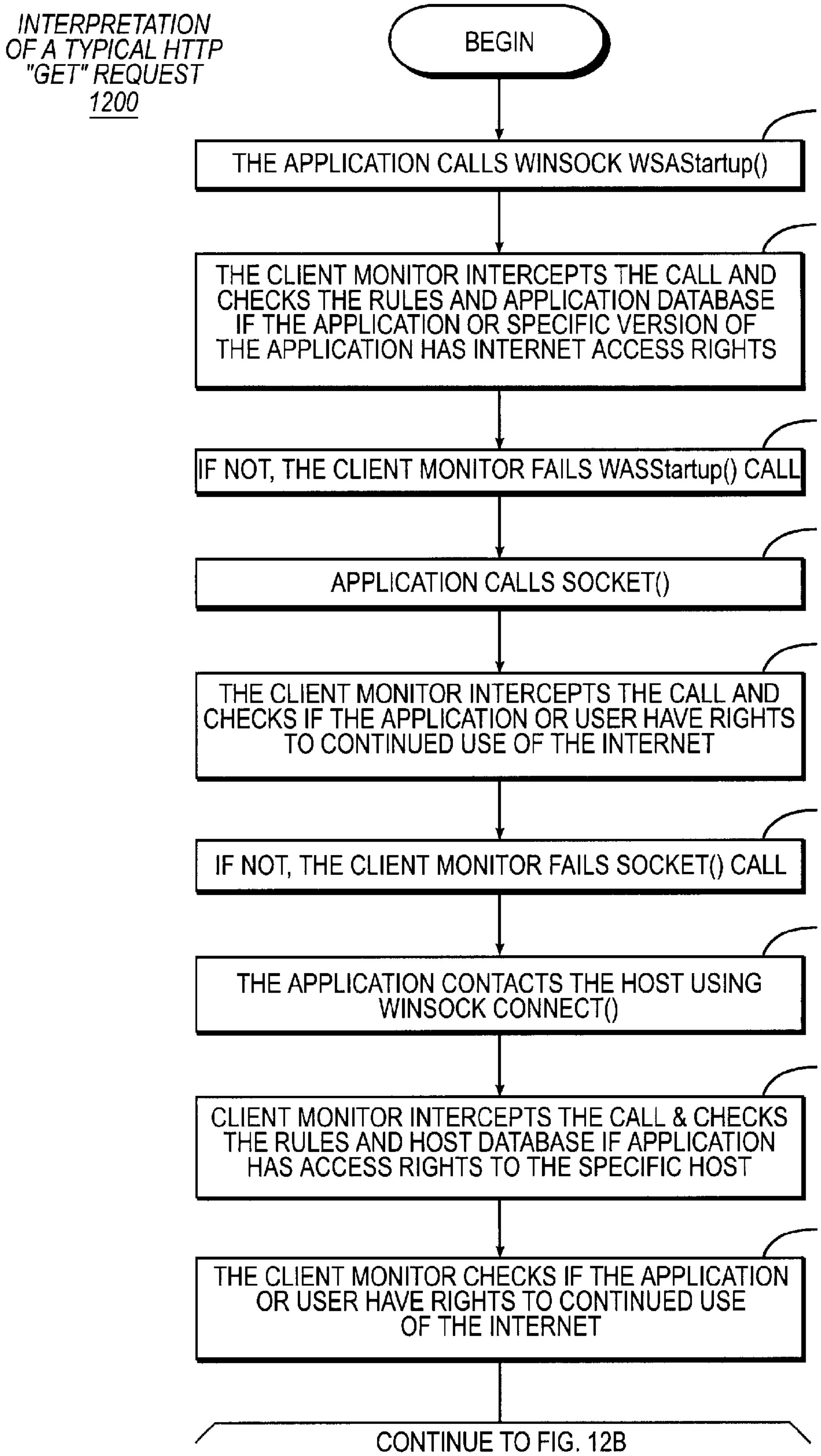

INTERPRETATION
OF A TYPICAL HTTP
"GET" REQUEST
1200
BEGIN
1201
THE APPLICATION CALLS WINSOCK WSAStartup()
1202
THE CLIENT MONITOR INTERCEPTS THE CALL AND CHECKS THE RULES AND APPLICATION DATABASE IF THE APPLICATION OR SPECIFIC VERSION OF THE APPLICATION HAS INTERNET ACCESS RIGHTS
1203
IF NOT, THE CLIENT MONITOR FAILS WASStartup() CALL
1204
APPLICATION CALLS SOCKET()
1205
THE CLIENT MONITOR INTERCEPTS THE CALL AND CHECKS IF THE APPLICATION OR USER HAVE RIGHTS TO CONTINUED USE OF THE INTERNET
1206
IF NOT, THE CLIENT MONITOR FAILS SOCKET() CALL
1207
THE APPLICATION CONTACTS THE HOST USING WINSOCK CONNECT()
1208
CLIENT MONITOR INTERCEPTS THE CALL & CHECKS THE RULES AND HOST DATABASE IF APPLICATION HAS ACCESS RIGHTS TO THE SPECIFIC HOST
1209
THE CLIENT MONITOR CHECKS IF THE APPLICATION OR USER HAVE RIGHTS TO CONTINUED USE OF THE INTERNET
CONTINUE TO FIG. 12B

FIG. 12A

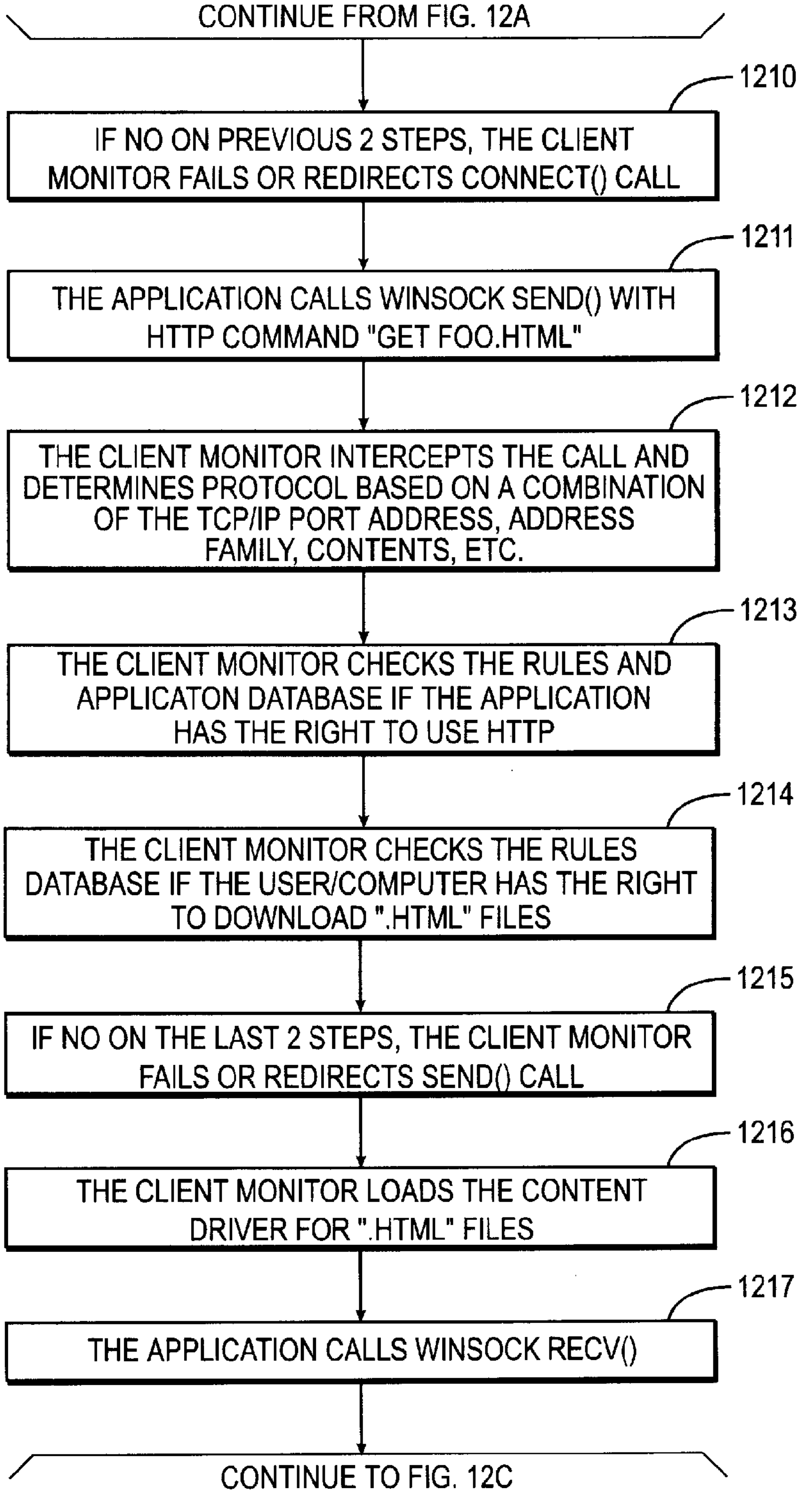
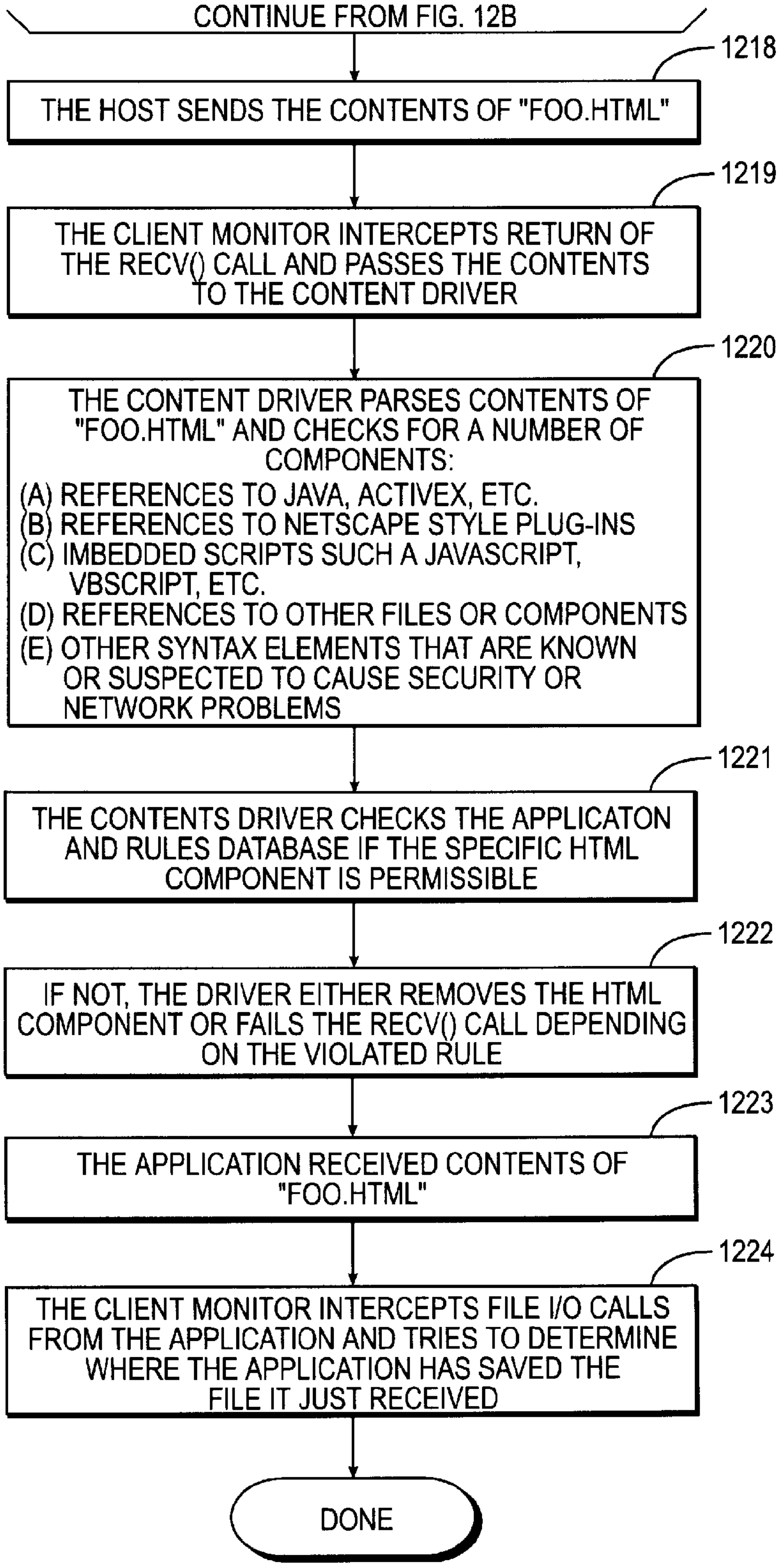

CONTINUE FROM FIG. 12A
1210
IF NO ON PREVIOUS 2 STEPS, THE CLIENT MONITOR FAILS OR REDIRECTS CONNECT() CALL
1211
THE APPLICATION CALLS WINSOCK SEND() WITH HTTP COMMAND "GET FOO.HTML"
1212
THE CLIENT MONITOR INTERCEPTS THE CALL AND DETERMINES PROTOCOL BASED ON A COMBINATION OF THE TCP/IP PORT ADDRESS, ADDRESS FAMILY, CONTENTS, ETC.
1213
THE CLIENT MONITOR CHECKS THE RULES AND APPLICATON DATABASE IF THE APPLICATION HAS THE RIGHT TO USE HTTP
1214
THE CLIENT MONITOR CHECKS THE RULES DATABASE IF THE USER/COMPUTER HAS THE RIGHT TO DOWNLOAD ".HTML" FILES
1215
IF NO ON THE LAST 2 STEPS, THE CLIENT MONITOR FAILS OR REDIRECTS SEND() CALL
1216
THE CLIENT MONITOR LOADS THE CONTENT DRIVER FOR ".HTML" FILES
1217
THE APPLICATION CALLS WINSOCK RECV()
CONTINUE TO FIG. 12C

FIG. 12B



THE HOST SENDS THE CONTENTS OF "FOO.HTML"

THE CLIENT MONITOR INTERCEPTS RETURN OF THE RECV() CALL AND PASSES THE CONTENTS TO THE CONTENT DRIVER

THE CONTENT DRIVER PARSES CONTENTS OF "FOO.HTML" AND CHECKS FOR A NUMBER OF COMPONENTS:
(A) REFERENCES TO JAVA, ACTIVEX, ETC.
(B) REFERENCES TO NETSCAPE STYLE PLUG-INS
(C) IMBEDDED SCRIPTS SUCH A JAVASCRIPT, VBSCRIPT, ETC.
(D) REFERENCES TO OTHER FILES OR COMPONENTS
(E) OTHER SYNTAX ELEMENTS THAT ARE KNOWN OR SUSPECTED TO CAUSE SECURITY OR NETWORK PROBLEMS

THE CONTENTS DRIVER CHECKS THE APPLICATON AND RULES DATABASE IF THE SPECIFIC HTML COMPONENT IS PERMISSIBLE

IF NOT, THE DRIVER EITHER REMOVES THE HTML COMPONENT OR FAILS THE RECV() CALL DEPENDING ON THE VIOLATED RULE

THE APPLICATION RECEIVED CONTENTS OF "FOO.HTML"

THE CLIENT MONITOR INTERCEPTS FILE I/O CALLS FROM THE APPLICATION AND TRIES TO DETERMINE WHERE THE APPLICATION HAS SAVED THE FILE IT JUST RECEIVED

FIG. 12C

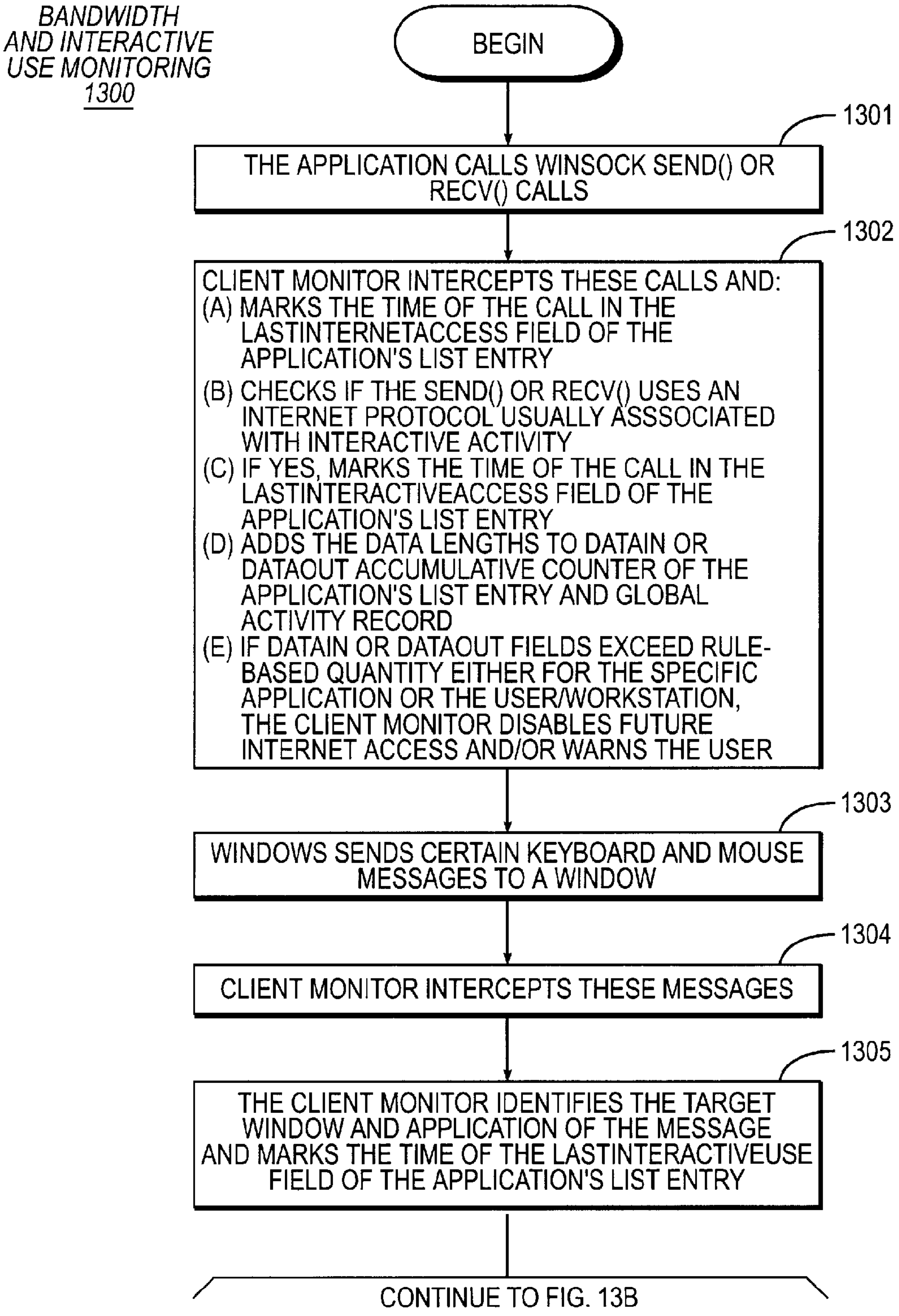

BANDWIDTH
AND INTERACTIVE
USE MONITORING
1300
BEGIN
1301
THE APPLICATION CALLS WINSOCK SEND() OR RECV() CALLS
1302
CLIENT MONITOR INTERCEPTS THESE CALLS AND:
(A) MARKS THE TIME OF THE CALL IN THE LASTINTERNETACCESS FIELD OF THE APPLICATION'S LIST ENTRY
(B) CHECKS IF THE SEND() OR RECV() USES AN INTERNET PROTOCOL USUALLY ASSSOCIATED WITH INTERACTIVE ACTIVITY
(C) IF YES, MARKS THE TIME OF THE CALL IN THE LASTINTERACTIVEACCESS FIELD OF THE APPLICATION'S LIST ENTRY
(D) ADDS THE DATA LENGTHS TO DATAIN OR DATAOUT ACCUMULATIVE COUNTER OF THE APPLICATION'S LIST ENTRY AND GLOBAL ACTIVITY RECORD
(E) IF DATAIN OR DATAOUT FIELDS EXCEED RULE-BASED QUANTITY EITHER FOR THE SPECIFIC APPLICATION OR THE USER/WORKSTATION, THE CLIENT MONITOR DISABLES FUTURE INTERNET ACCESS AND/OR WARNS THE USER
1303
WINDOWS SENDS CERTAIN KEYBOARD AND MOUSE MESSAGES TO A WINDOW
1304
CLIENT MONITOR INTERCEPTS THESE MESSAGES
1305
THE CLIENT MONITOR IDENTIFIES THE TARGET WINDOW AND APPLICATION OF THE MESSAGE AND MARKS THE TIME OF THE LASTINTERACTIVEUSE FIELD OF THE APPLICATION'S LIST ENTRY
CONTINUE TO FIG. 13B

FIG. 13A

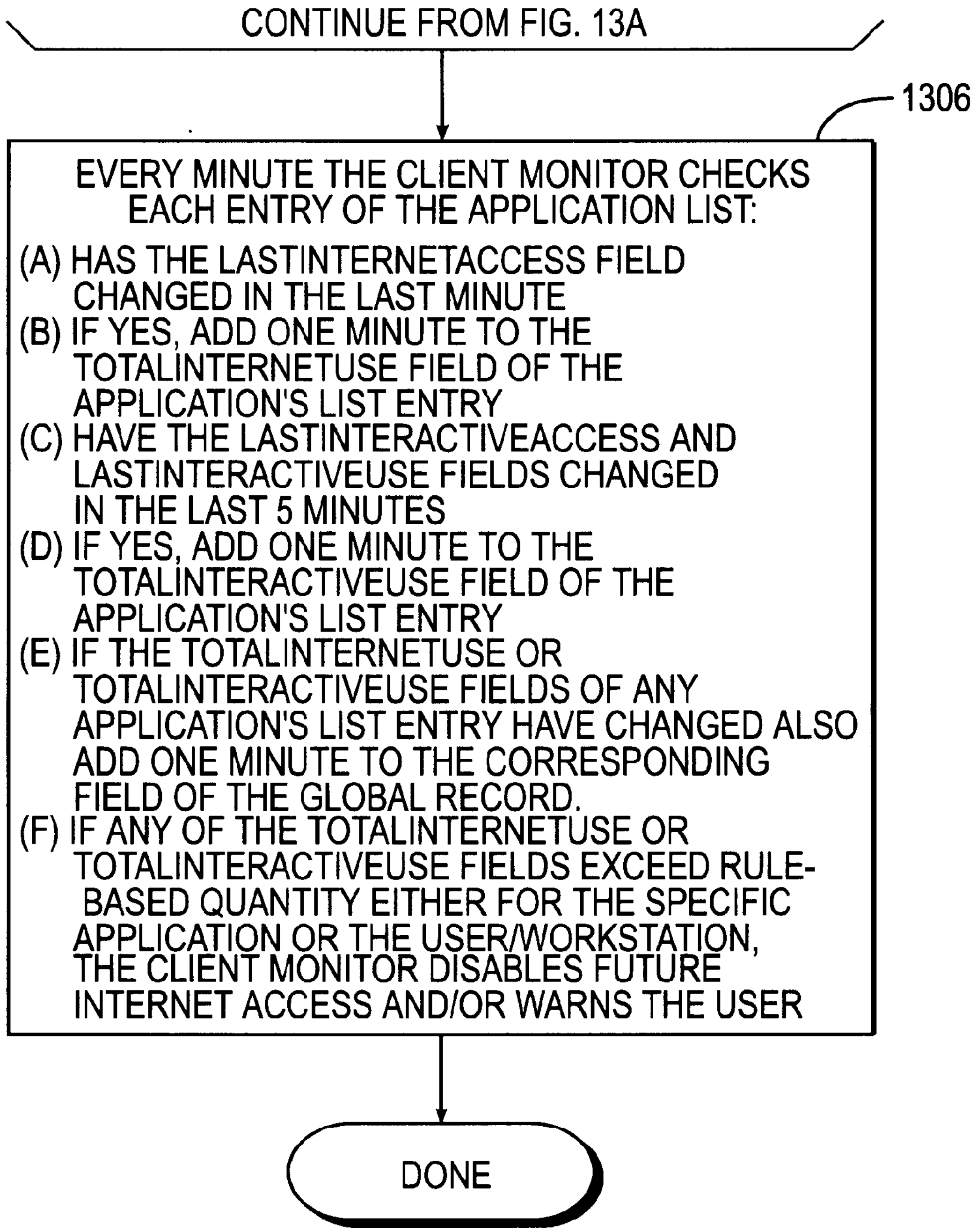
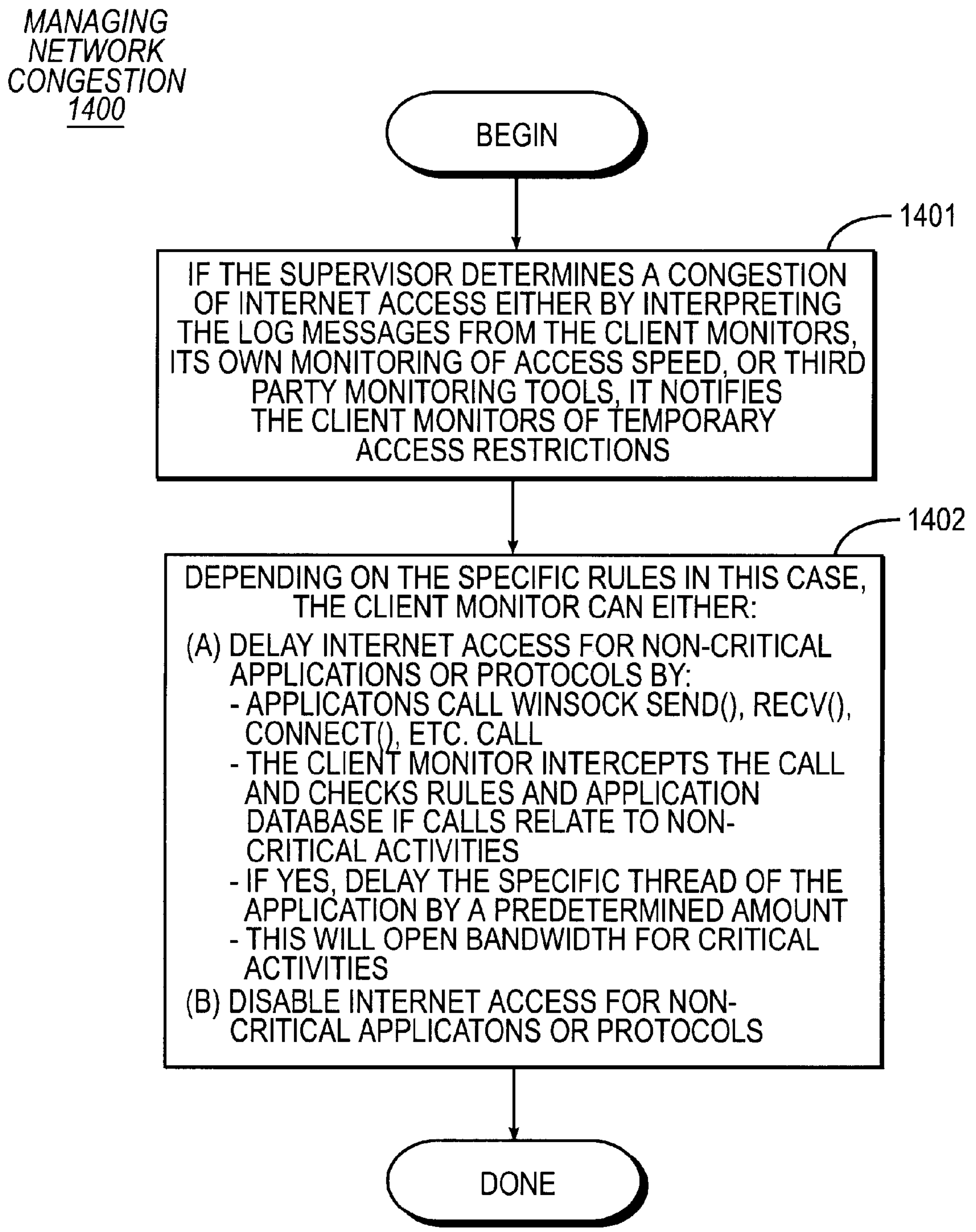

CONTINUE FROM FIG. 13A
1306
EVERY MINUTE THE CLIENT MONITOR CHECKS EACH ENTRY OF THE APPLICATION LIST:
(A) HAS THE LASTINTERNETACCESS FIELD CHANGED IN THE LAST MINUTE
(B) IF YES, ADD ONE MINUTE TO THE TOTALINTERNETUSE FIELD OF THE APPLICATION'S LIST ENTRY
(C) HAVE THE LASTINTERACTIVEACCESS AND LASTINTERACTIVEUSE FIELDS CHANGED IN THE LAST 5 MINUTES
(D) IF YES, ADD ONE MINUTE TO THE TOTALINTERACTIVEUSE FIELD OF THE APPLICATION'S LIST ENTRY
(E) IF THE TOTALINTERNETUSE OR TOTALINTERACTIVEUSE FIELDS OF ANY APPLICATION'S LIST ENTRY HAVE CHANGED ALSO ADD ONE MINUTE TO THE CORRESPONDING FIELD OF THE GLOBAL RECORD.
(F) IF ANY OF THE TOTALINTERNETUSE OR TOTALINTERACTIVEUSE FIELDS EXCEED RULE-BASED QUANTITY EITHER FOR THE SPECIFIC APPLICATION OR THE USER/WORKSTATION, THE CLIENT MONITOR DISABLES FUTURE INTERNET ACCESS AND/OR WARNS THE USER
DONE

FIG. 13B



IF THE SUPERVISOR DETERMINES A CONGESTION OF INTERNET ACCESS EITHER BY INTERPRETING THE LOG MESSAGES FROM THE CLIENT MONITORS, ITS OWN MONITORING OF ACCESS SPEED, OR THIRD PARTY MONITORING TOOLS, IT NOTIFIES THE CLIENT MONITORS OF TEMPORARY ACCESS RESTRICTIONS

DEPENDING ON THE SPECIFIC RULES IN THIS CASE, THE CLIENT MONITOR CAN EITHER:

(A) DELAY INTERNET ACCESS FOR NON-CRITICAL APPLICATIONS OR PROTOCOLS BY:
- APPLICATONS CALL WINSOCK SEND(), RECV(), CONNECT(), ETC. CALL
- THE CLIENT MONITOR INTERCEPTS THE CALL AND CHECKS RULES AND APPLICATION DATABASE IF CALLS RELATE TO NON-CRITICAL ACTIVITIES
- IF YES, DELAY THE SPECIFIC THREAD OF THE APPLICATION BY A PREDETERMINED AMOUNT
- THIS WILL OPEN BANDWIDTH FOR CRITICAL ACTIVITIES

(B) DISABLE INTERNET ACCESS FOR NON-CRITICAL APPLICATONS OR PROTOCOLS

FIG. 14

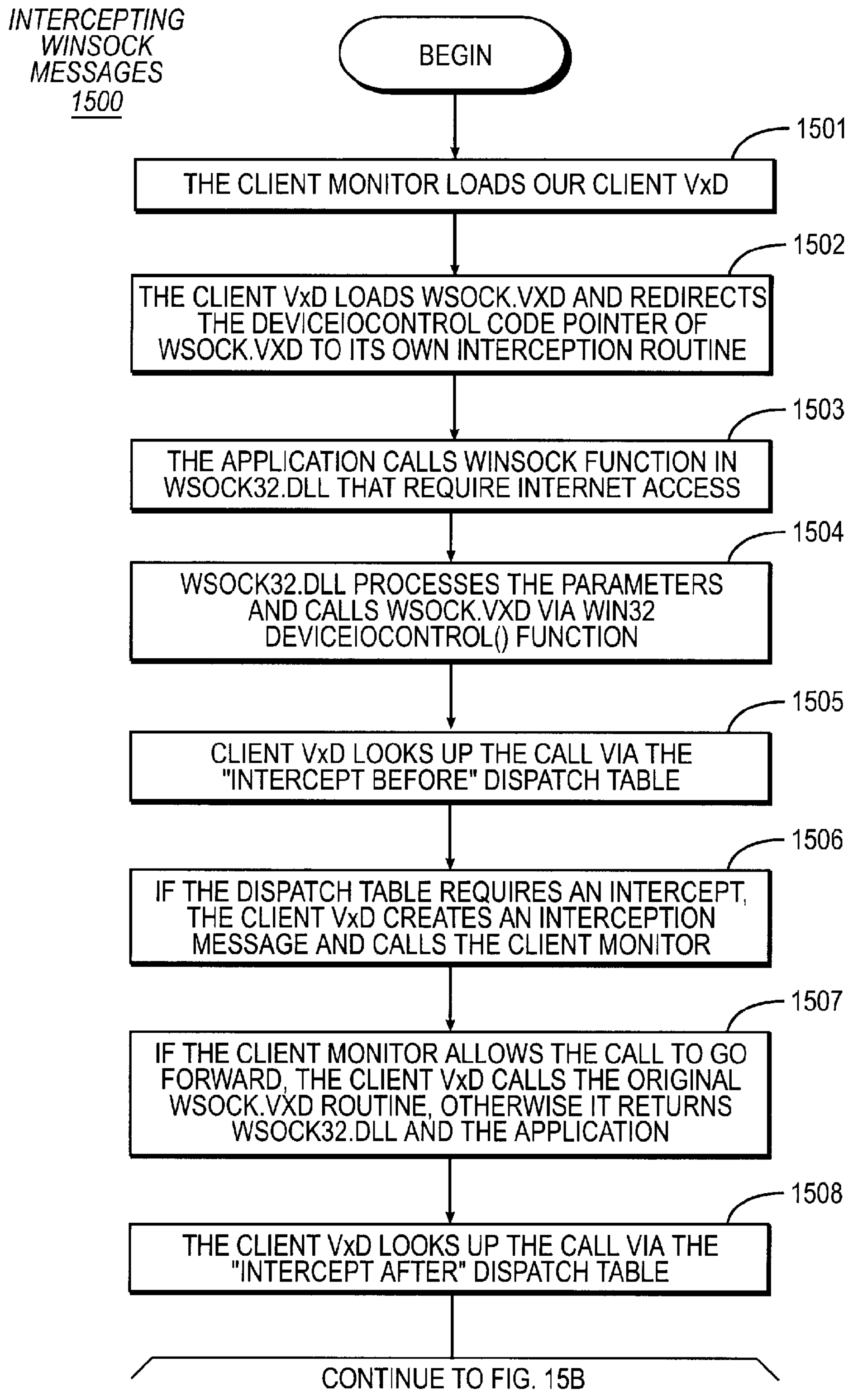

INTERCEPTING WINSOCK MESSAGES 1500
BEGIN
1501
THE CLIENT MONITOR LOADS OUR CLIENT VxD
1502
THE CLIENT VxD LOADS WSOCK.VXD AND REDIRECTS THE DEVICEIOCONTROL CODE POINTER OF WSOCK.VXD TO ITS OWN INTERCEPTION ROUTINE
1503
THE APPLICATION CALLS WINSOCK FUNCTION IN WSOCK32.DLL THAT REQUIRE INTERNET ACCESS
1504
WSOCK32.DLL PROCESSES THE PARAMETERS AND CALLS WSOCK.VXD VIA WIN32 DEVICEIOCONTROL() FUNCTION
1505
CLIENT VxD LOOKS UP THE CALL VIA THE "INTERCEPT BEFORE" DISPATCH TABLE
1506
IF THE DISPATCH TABLE REQUIRES AN INTERCEPT, THE CLIENT VxD CREATES AN INTERCEPTION MESSAGE AND CALLS THE CLIENT MONITOR
1507
IF THE CLIENT MONITOR ALLOWS THE CALL TO GO FORWARD, THE CLIENT VxD CALLS THE ORIGINAL WSOCK.VXD ROUTINE, OTHERWISE IT RETURNS WSOCK32.DLL AND THE APPLICATION
1508
THE CLIENT VxD LOOKS UP THE CALL VIA THE "INTERCEPT AFTER" DISPATCH TABLE
CONTINUE TO FIG. 15B

FIG. 15A

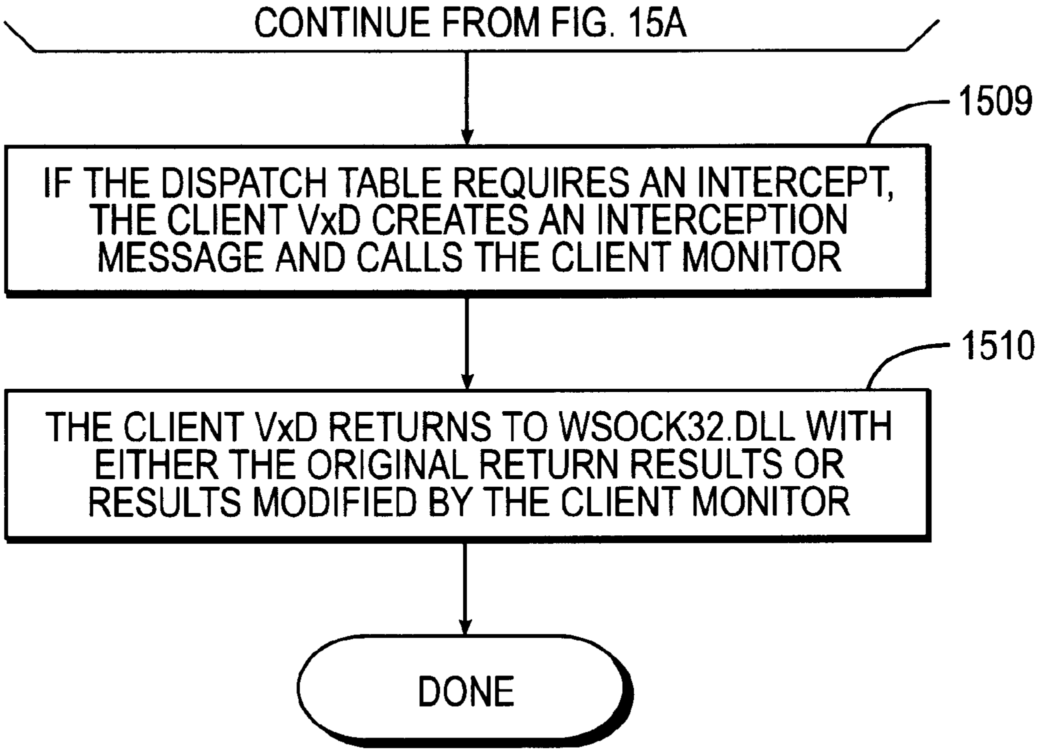

CONTINUE FROM FIG. 15A
1509
IF THE DISPATCH TABLE REQUIRES AN INTERCEPT, THE CLIENT VxD CREATES AN INTERCEPTION MESSAGE AND CALLS THE CLIENT MONITOR
1510
THE CLIENT VxD RETURNS TO WSOCK32.DLL WITH EITHER THE ORIGINAL RETURN RESULTS OR RESULTS MODIFIED BY THE CLIENT MONITOR
DONE

FIG. 15B

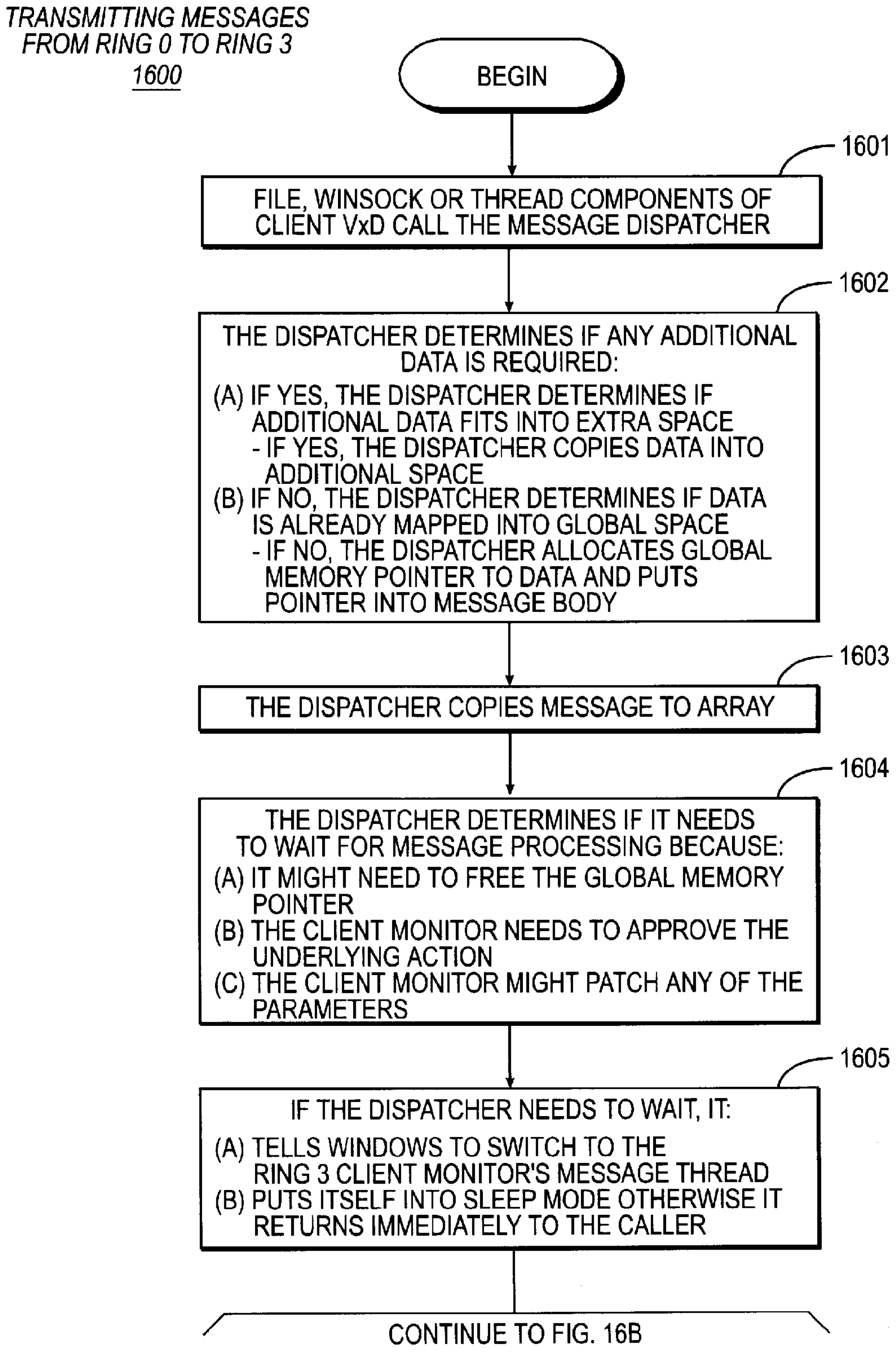

TRANSMITTING MESSAGES FROM RING 0 TO RING 3
1600
BEGIN
1601
FILE, WINSOCK OR THREAD COMPONENTS OF CLIENT VxD CALL THE MESSAGE DISPATCHER
1602
THE DISPATCHER DETERMINES IF ANY ADDITIONAL DATA IS REQUIRED:
(A) IF YES, THE DISPATCHER DETERMINES IF ADDITIONAL DATA FITS INTO EXTRA SPACE
- IF YES, THE DISPATCHER COPIES DATA INTO ADDITIONAL SPACE
(B) IF NO, THE DISPATCHER DETERMINES IF DATA IS ALREADY MAPPED INTO GLOBAL SPACE
- IF NO, THE DISPATCHER ALLOCATES GLOBAL MEMORY POINTER TO DATA AND PUTS POINTER INTO MESSAGE BODY
1603
THE DISPATCHER COPIES MESSAGE TO ARRAY
1604
THE DISPATCHER DETERMINES IF IT NEEDS TO WAIT FOR MESSAGE PROCESSING BECAUSE:
(A) IT MIGHT NEED TO FREE THE GLOBAL MEMORY POINTER
(B) THE CLIENT MONITOR NEEDS TO APPROVE THE UNDERLYING ACTION
(C) THE CLIENT MONITOR MIGHT PATCH ANY OF THE PARAMETERS
1605
IF THE DISPATCHER NEEDS TO WAIT, IT:
(A) TELLS WINDOWS TO SWITCH TO THE RING 3 CLIENT MONITOR'S MESSAGE THREAD
(B) PUTS ITSELF INTO SLEEP MODE OTHERWISE IT RETURNS IMMEDIATELY TO THE CALLER
CONTINUE TO FIG. 16B

FIG. 16A

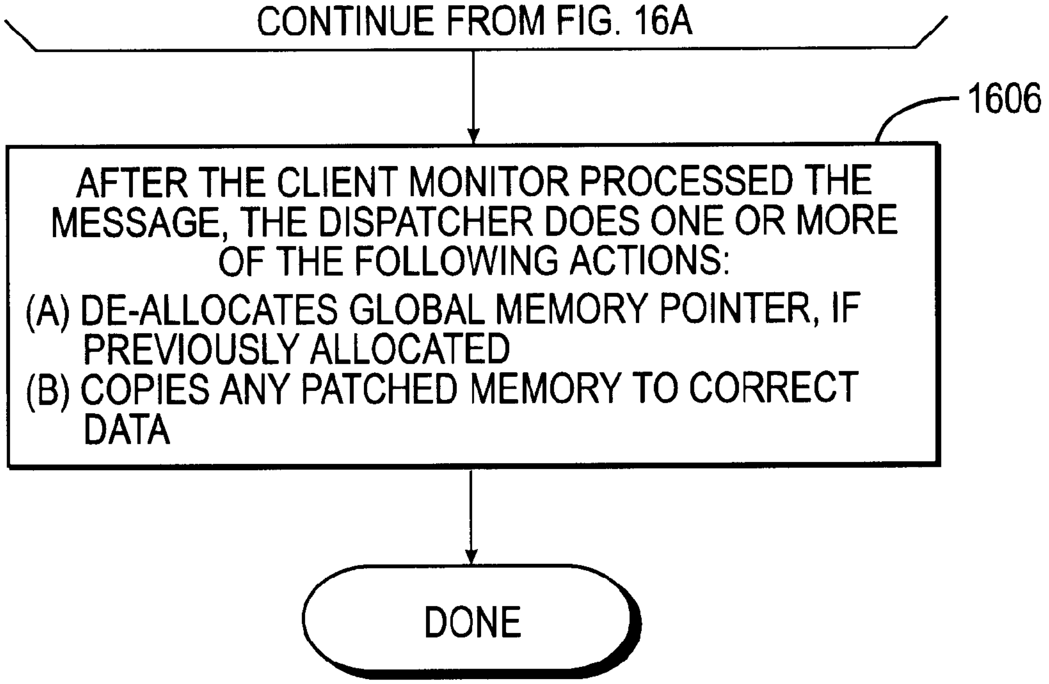

CONTINUE FROM FIG. 16A
1606
AFTER THE CLIENT MONITOR PROCESSED THE MESSAGE, THE DISPATCHER DOES ONE OR MORE OF THE FOLLOWING ACTIONS:
(A) DE-ALLOCATES GLOBAL MEMORY POINTER, IF PREVIOUSLY ALLOCATED
(B) COPIES ANY PATCHED MEMORY TO CORRECT DATA
DONE

FIG. 16B

SYSTEM AND METHODOLOGY FOR MANAGING INTERNET ACCESS ON A PER APPLICATION BASIS FOR CLIENT COMPUTERS CONNECTED TO THE INTERNET

The present application claims priority from commonly-owned provisional patent application Ser. No. 60/033,975, filed Dec. 31, 1996, entitled SYSTEM AND METHODS FOR MONITORING INTERNET ACCESS, and listing as inventor Gregor P. Freund, the disclosure of which is hereby incorporated by reference.

COPYRIGHT NOTICE

BACKGROUND OF THE INVENTION

The present invention relates generally to information processing and, more particularly, to system and methods for regulating access and maintaining security of individual computer systems and local area networks (LANs) connected to larger open networks (Wide Area Networks or WANs), including the Internet.

The first personal computers were largely stand-alone units with no direct connection to other computers or computer networks. Data exchanges between computers were mainly accomplished by exchanging magnetic or optical media such as floppy disks. Over time, more and more computers were connected to each other using Local Area Networks or "LANs." In both cases, maintaining security and controlling what information a user of a personal computer can access was relatively simple because the overall computing environment was limited and clearly defined.

With the ever-increasing popularity of the Internet, particularly the World Wide Web ("Web") portion of the Internet, however, more and more personal computers are connected to larger networks. Providing access to vast stores of information, the Internet is typically accessed by users through Web "browsers" (e.g., Microsoft Internet Explorer™ or Netscape Navigator™ browser software) or other "Internet applications." Browsers and other Internet applications include the ability to access a URL (Universal Resource Locator) or "Web" site. The explosive growth of the Internet had a dramatic effect on the LANs of many businesses and other organizations. More and more employees need direct access through their corporate LAN to the Internet in order to facilitate research, competitive analysis, communication between branch offices, and send e-mail, to name just a few.

As a result, corporate IS (Information Systems) departments now face unprecedented challenges. Specifically, such departments, which have to date operated largely in a clearly defined and friendly environment, are now confronted with a far more complicated and hostile situation. As more and more computers are now connected to the Internet, either directly (e.g., over a dialup connection with an Internet Service Provider or "ISP") or through a gateway between a LAN and the Internet, a whole new set of challenges face LAN administrators and individual users alike: these previously-closed computing environments are now opened to a worldwide network of computer systems. Specific challenges, for example, include the following: (1) attacks by perpetrators (hackers) capable of damaging the local computer systems, misuse these systems, or steal proprietary data and programs; (2) unauthorized access to external data (e.g., pornographic or other unsuitable Web sites); (3) infiltration by viruses and "Trojan Horse" programs; (4) abuse of the local computer system for unauthorized personal activities (e.g., extensive Web browsing or game playing) with subsequent loss of productivity; and (5) hording available network bandwidth through use of bandwidth-intensive applications (e.g., real-time audio programs).

The software industry has, in response, introduced a myriad of products and technologies to address and minimize these threats, including "firewalls," proxy servers, and similar technologies—all designed to keep outside hackers from penetrating the LAN. Firewalls are applications that intercept the data traffic at the gateway to a wide area network (WAN) and try to check the data packets (i.e., Internet Protocol packets or "IP packets") being exchanged for suspicious or unwanted activities. Initially firewalls have been used primarily to keep intruders from the LAN by filtering data packets. More recently, the concept has been expanded to include "Stateful Inspection." Here, a firewall not only looks at the IP packets but also inspects the data packets transport protocol (e.g., TCP) header (and even the application level protocols) in an attempt to better understand the exact nature of the data exchange.

Proxy server or Application Gateways, on the other hand, are LAN server-based applications that act on behalf of the client application. Accessing the Internet directly, the application first submits a request to the proxy server which inspects the request for unsafe or unwanted traffic. Only after this inspection will the proxy server consider forwarding the request to the destination on the Internet.

Both strategies are based on a centralized filter mechanism, with most of the filtering work being performed at the server (as opposed to the individual client PCs). Such an approach is problematic, however. Because of the centralized nature of firewalls and proxy servers, each approach extracts significant performance penalties. During operation of a typical system employing either approach, a single server might have to do the filtering work for hundreds or even thousands of PCs or workstations. This represents a major bottleneck to overall system performance. Further, a centralized filter poses a significant bottleneck even when client PCs are idly awaiting data. As emerging technologies on the Internet require still faster data delivery (e.g., real-time audio and video fees) and use more complex protocols, this problem will likely be exacerbated. In the case of firewalls employing "Stateful Inspection" technology, performance problems are aggravated by the fact that the firewall software needs to duplicate much of the protocol implementation of the client application as well as the transport protocol (e.g., TCP and UDP protocol) in order to understand the data flow.

As another problem, centralized filter architectures are missing vital information to correctly interpret the data packets because the underlying protocols were designed for effective data transfer and not for data monitoring and interception. For instance, monitoring based on an individual client application (or versions thereof) is not supported, all despite the fact that two identical data packets (or series of data packets) can have completely different meanings based on the underlying context—that is, how the client application actually interprets the data packets. As a result, computer viruses or Trojan Horse applications can camouflage data transmissions as legitimate traffic.

There are still other disadvantages to centralized filtering. The approach is difficult to configure and administer. The task of setting up different rights for different users, workstations, or workgroups, for instance, is particularly difficult. No facilities are provided for delegating certain access and monitoring authority, for example, in order to allow a workgroup supervisor to manage less critical aspects of the Internet access for his or her group without going through a central authority. Also, a centralized filter cannot distinguish between "active" use of the Internet (i.e., when user interaction with the PC causes the Internet access) and "background" use (i.e., when an application accesses the Internet without user interaction). Still further, a centralized filter is easily circumvented, for example by a user employing a modem for establishing a dial-up connection to an ISP (Internet Service Provider). Similarly, the proxy-server approach is unattractive. Special versions or specialized configurations of client applications are required, thus complicating system administration. Internet setup for portable computers employed at remote locations is especially complicated.

Providing a client-based filter (e.g., SurfWatch and CyberPatrol) for preventing users from accessing undesirable World Wide Web sites does not adequately overcome the disadvantages of centralized filtering. Designed largely as parental control tools for individual PCs, these programs are easily disabled by uninstalling (accidentally or intentionally) the filter. A Windows user can, for example, simply reinstall Windows, replacing certain driver files of the filter. This disables the filter and provides the user with unrestricted access to the Internet.

All told, comparably little has been done to date to effectively minimize or eliminate the risks posed from within one's own corporate LAN, specifically, how one manages access to the Internet or other WAN from client machines. Quite simply, the technical framework to successfully implement an Internet access management product does not exists. What is needed are system and methods providing network administrators, workgroup supervisor, and individual PC users with the ability to monitor and regulate the kinds of exchanges permissible between one's local computing environment and external network or WANs, including the Internet. The present invention fulfills this and other needs.

SUMMARY OF THE INVENTION

The present invention provides system and methods for client-based monitoring and filtering of access, which operates in conjunction with a centralized enforcement supervisor. In accordance with the present invention, a central filter is not employed. Instead, the present invention provides a client-side filter that is controlled by the centralized authority as long as the centralized authority has a way of enforcing non-compliance, for example, by blocking access to an open network, such as a WAN or the Internet.

At a general level, the present invention provides a system comprising one or more access management applications that set access rules for the entire LAN for one or more workgroups or individual users, a client-based filter application (installed at each client), and a central supervisor application that maintains the access rules for the client based filter and verifies the existence and proper operation of the client-based filter application. Typically, the system includes (optionally) a firewall or similar application, which works together with the supervisor application in order to block all clients that have not been verified by the supervisor application.

The access management application is employed by the LAN administrator, workgroup administrator, and/or LAN user to maintain a database of the access rules for the workstations being administrated. These access rules can include criteria such as total time a user can be connected to the Internet (e.g., per day, week, month, or the like), time a user can interactively use the Internet (e.g., per day, week, month, or the like), a list of applications or application versions that a user can or cannot use in order to access the Internet, a list of URLs (or WAN addresses) that a user application can (or cannot) access, a list of protocols or protocol components (such as Java Script™) that a user application can or cannot use, and rules to determine what events should be logged (including how long are logs to be kept). These access rules can be qualified by optionally specifying: to whom should a rule apply (list of users, list of workgroups, or all); start date and expiration date of a rule; time of day when the rule should be applied (for example from 9 am to 5 pm); whether the rule is "disclosed" to the user or workgroup manager or remains hidden; whether a rule can be overwritten or modified by the workgroup manager or user; and what should happen if a rule is violated (e.g., denying Internet access, issue a warning, redirecting the access, creating a log entry, or the like).

The client-based filter application, which in a preferred embodiment performs all of the monitoring, logging, and filtering work, is responsible for intercepting process loading and unloading. Other responsibilities include keeping a list of currently active processes; intercepting certain keyboard, mouse and other interactive user activities in order to determine which process is actively used; intercepting and interpreting all TCP/IP communication and build a comprehensive representation of these TCP/IP activities; and intercepting certain file activity and assign them to the originating process.

By intercepting process loading and unloading and keeping a list of currently active processes, each client process can be checked for various characteristics, including checking executable name, version numbers, executable file checksums, version header details, configuration settings, and the like. With this information, the system can determine if the process in question should have access to the Internet and what kind of access (i.e., protocols, Internet addresses, time limitations, and the like) is permissible for the given specific user.

By intercepting and interpreting all TCP/IP communication and building a comprehensive representation of these TCP/IP activities, the system can monitor TCP/IP activities on a per process or per application basis. If a particular process has access rights to the Internet (and is permitted to use the detected protocol and no other rules are violated), the communication of the process is logged and allowed to go forward. Otherwise, the prescribed remedial action for any violated rule is performed, including logging an exception log entry and, depending on the rules the TCP/IP activity, the communication is either terminated, redirected, modified, or continued. In a similar fashion, any possible time limitation rules are evaluated and enforced at this point.

By intercepting certain file activity and assigning them to the originating process, the system can track files being created and changed by any process in order to match TCP/IP activities with corresponding file activities. If a process uses FTP to download a file, for example, the system will match that activity to a file being saved by the same process by checking file name and size. If a match is found, a log entry is generated. This allows the immediate application of internal or external virus checkers.

The centralized supervisor application is installed on a computer on the LAN that can be reached from all workstations that need access to the Internet; this is typically (although not necessarily) a server computer. The supervisor monitors whether a client has the filter application loaded and provides the filter application with the rules for the specific user or workstation. The filter application maintains a local copy of these rules so that rule enforcement continues even when the user accesses the Internet but bypasses the LAN (e.g., a mobile computer on the road). The communication between the client-based filter and the centralized supervisor application, as well as between the supervisor application and the firewall, employs encryption to ensure secure communication and avoid any possible attack on that level.

The system of the present invention works together with existing firewalls which allow a program (e.g., the supervisor application) to dynamically set the addresses of the workstations that should have access to the Internet. The supervisor application signals the firewall which client applications have been "certified" so that the firewall only grants Internet access to those clients. At the same time, a firewall can continue to perform its usual duties, such as protecting the LAN from outside intruders or protecting the LAN and server operating system(s).

Exemplary methodologies of the present invention include the following.

I. Client Monitor with Supervisor/Firewall Backup and Enforcement

a) Installing at a particular client computer a client monitoring process;

b) Installing at another computer on the local area network a supervisor process, which specifies rules which govern Internet access by the client computers including the particular client computer;

c) Transmitting a filtered subset of the rules to the particular client computer;

d) At the client monitoring process, trapping a request for Internet access from the particular client computer;

e) Determining whether the request for Internet access would violate any of the rules transmitted to the particular client computer, and

f) If the request for Internet access violates any of the rules transmitted to the particular client computer, denying the request for Internet access.

II. Using Application Properties to Determine Legitimate Internet Traffic

a) Application attempts to access Internet;

b) Client Monitor compares application properties (version, executable name, and the like) with database of application allowed to access the Internet and checks what kind of activity the application is allowed to do (mail, browsing, and the like); and

c) If application is not allowed to access the Internet or not allowed to use the specific protocol then client monitor can stop application from accessing the Internet and/or warn user.

III. Using Application Properties to Determine if an Applications has Known Security Flaws

a) Application attempts to access Internet;

b) Client Monitor compares application properties (version, executable name, and the like) with database of application with known security problems; and

c) If application has know security problems, client monitor stops the application from accessing the Internet and/or warns the user.

IV. Monitoring User Interaction (e.g., keyboard/mouse and the like) to Distinguish and Regulate Time Spent Online;

a) Client Monitor detects interactive commands (e.g., keyboard/mouse) for an application that uses the Internet via "browsing" protocols (e.g., HTTP);

b) Client monitor determines whether the user interactively uses the Internet and restrict the activity if required.

V. Using Client Monitor to Alleviate Network Congestion

a) Supervisor Application notifies client that network is congested; and

b) Client Monitor delays transmission of non-time critical information and data.

VI. Using Local and Remote Stored Rules Databases to Allow Client Monitor Functioning Even if Supervisor Application is Not Available

a) Client monitor attempts but is unable to access the supervisor application; and

b) Access rules are still enforced because Client Monitor employs a local copy of rules (previously downloaded).

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram illustrating a computer system in which the present invention may be embodied.

FIG. **2** is a block diagram of a computer software system for controlling the operation of the computer system of FIG. **1**.

FIGS. **3**A–B are block diagrams providing an overview of Internet-based (client/server) systems in which the present invention may be embodied.

FIG. **4** is a block diagram illustrating client-side operation of the system.

FIG. **5** is a block diagram illustrating a client-side monitor or data acquisition module.

FIGS. **6**A–E are bitmap screenshots illustrating a preferred user interface provided by the client-side monitoring component of the present invention.

FIGS. **7**A–K are bitmap screenshots illustrating a preferred user interface or "wizard" dialog for configuring rules.

FIGS. **8**A–B comprise a flowchart illustrating a method of the present invention for loading the Client Monitor component.

FIG. **9** is a flowchart illustrating a method of the present invention for handling the scenario of when the Client Monitor is unable to locate a Supervisor.

FIG. **10** is a flowchart illustrating a method of the present invention for unloading the Client Monitor component.

FIGS. **11**A–B comprise a flowchart illustrating a method of the present invention for loading the Client Monitor in an Internet Service Provider (ISP) environment.

FIGS. **12**A–C comprise a flowchart illustrating a method of the present invention for interpreting protocol commands, such as a typical HTTP "GET" request.

FIGS. **13**A–B comprise a flowchart illustrating a method of the present invention for bandwidth and interactive use monitoring.

FIG. **14** is a flowchart illustrating a method of the present invention for managing network congestion.

FIGS. **15**A–B comprise a flowchart illustrating a method of the present invention for intercepting communication driver (e.g., WinSock) messages.

FIGS. **16**A–B comprise a flowchart illustrating a method of the present invention for transmitting messages from one memory protection ring to another (e.g., from highly-privileged Ring **0** to lesser-privileged Ring **3**).

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

The following description will focus on the presently-preferred embodiment of the present invention, which is operative in an Internet-connected environment, including, for instance, client machines running under the Microsoft® Windows environment and connected to an open network, such as a WAN or the Internet. The present invention, however, is not limited to any particular one application or any particular environment. Instead, those skilled in the art will find that the system and methods of the present invention may be advantageously applied to a variety of system and application software, including database management systems, word processors, spreadsheets, and the like, operating on a variety of different platforms, including the Macintosh® operating system, the UNIX® operating system, NextStep® operating system, and the like. Therefore, the description of the exemplary embodiments which follows is for purposes of illustration and not limitation.

System Architecture

A. System hardware (for client and/or server machines)

The invention is generally embodied on a computer system including one or more computer systems, such as computer system **100** of FIG. **1**, operating on a network. System **100** comprises a central processor **101**, a main memory **102**, an input/output controller **103**, a keyboard **104**, a pointing device **105** (e.g., mouse, track ball, pen device, or the like), a display or screen device **106**, and a mass storage **107** (e.g., hard or fixed disk, removable floppy disk, optical disk, magneto-optical disk, or flash memory), a network interface card or controller **111** (e.g., Ethernet), and a modem **112** (e.g., 28.8K baud modem or ISDN modem). Although not shown separately, a real-time system clock is included with the system **100**, in a conventional manner. Processor **101** includes or is coupled to a cache memory **109** for storing frequently accessed information; memory **109** may be an on-chip cache or external cache (as shown). One or more input/output device(s) **108**, such as a printing device or slide output device, are included in the system **100**, as desired. As shown, the various components of the system **100** communicate through a system bus **110** or similar architecture. The system itself communicates with other systems via a network interface card **111** (e.g., available from 3Com) and/or modem **112** (e.g., available from U.S. Robotics). In a preferred embodiment, the system **100** includes an IBM PC-compatible personal computer, available from a variety of vendors (including IBM of Armonk, N.Y.). I/O device **108** may include a laser printer, such as an HP Laserjet printer, which is available from Hewlett-Packard of Palo Alto, Calif.

B. System software (for controlling clients and server machines)

Illustrated in FIG. **2**, a computer software system **220** is provided for directing the operation of the computer system **100**. Software system **220**, which is stored in system memory **102** and on storage (e.g., disk memory) **107**, includes a kernel or operating system (OS) **240** and a windows shell **250**. One or more application programs, such as client application software or "programs" **245** may be "loaded" (i.e., transferred from storage **107** into memory **102**) for execution by the system **100**. In a preferred embodiment, client application software includes a Web browser (e.g., Netscape Navigator™ or Microsoft Internet Explorer™ browser software) which communicates through a communication layer or driver **241** (e.g., Winsock) with the Internet.

System **220** includes a user interface (UI) **260**, preferably a Graphical User Interface (GUI), for receiving user commands and data. These inputs, in turn, may be acted upon by the system **100** in accordance with instructions from operating module **240**, windows **250**, and/or client application module(s) **245**. The UI **260** also serves to display the results of operation from the OS **240**, windows **250**, and application (s) **245**, whereupon the user may supply additional inputs or terminate the session. OS **240** and windows **245** can be provided by Microsoft® Windows 95, by Microsoft® Windows NT, or by Microsoft® Windows 3.x (operating in conjunction with MS-DOS); these are available from Microsoft Corporation of Redmond, Wash. Alternatively, OS **240** and windows **245** can be provided by IBM OS/2® (available from IBM of Armonk, N.Y.) or Macintosh® OS (available from Apple Computers of Cupertino, Calif.). Although shown conceptually as a separate module, the UI is typically provided by interaction of the application modules with the windows shell, both operating under OS **240**.

Of particular interest, the system **220** includes a client-side Internet access monitoring module **225** of the present invention. Internet access monitoring module **225** interfaces directly with Winsock driver **241** via a Windows VxD driver interface surfaced by the driver **241**, as shown at **243**. Construction and operation of the client-side Internet access monitoring module **225**, including its interaction with server-based components employed in a preferred embodiment, will now be described in further detail.

Preferred Monitoring and Management of Internet Access

A. Introduction

1. General design

An Internet access monitoring system, constructed in accordance with the present invention, preferably includes the following.

(1) The system should preferably be capable of restricting access to the Internet (or other Wide Area Network) to certain approved applications or/and application versions.

(2) The system should preferably support centrally-maintained access rules (e.g., defining basic access rights), but at the same time allow individual work-group managers or even individual users to set rules for their area of responsibility, if so desired by the organization.

(3) The system should preferably prevent users from circumventing Internet access rules, either accidentally or intentionally. It should be difficult, for instance, for a user to circumvent access rules by connecting to the Internet through a dial-up connection (e.g., connecting to an ISP with a modem). Similarly, it should be difficult for a user to circumvent access rules by uninstalling or tampering with components of the system, from his/her own PC.

(4) The system should preferably regulate the amount of time users can access the Internet, with specific considerations for detecting interactive or "active" use by the user versus background or "passive" use by certain applications. This differentiation prevents the system

from mistakenly blocking access of a user based on excessive use, when in fact a background application is instead responsible for the activity.

(5) The system should preferably be capable of restricting individual users, workgroups, or an entire organization to undertaking certain permissible on-line activities, such as allowing use of Web browsing programs but disallowing use of Web radio programs (e.g., RealAudio™).

(6) The system should preferably be capable of restricting individual users, workgroups, or an entire organization to accessing certain external computer sites or to prohibit access to a specific list of sites.

(7) The system should preferably be capable of filtering incoming data, including binary files, for detecting viruses and Trojan Horse programs.

(8) The system should preferably maintain a detailed log of access related activities, for facilitating administration needs.

2. Managing Internet access along organizational structures

In order to effectively manage Internet access, a system should support existing organizational structures. A department supervisor, for example, should be able to determine the needs of his/her subordinates within a safe overall framework. This is important for the overall success of Internet access within the organization as it allows supervisors to address any problems which arise early on (before they become serious personnel issues). Accordingly, the Internet access monitoring system of the present invention supports a hierarchical structure where individual supervisors can monitor and set the access rules for their individual workgroups without affecting others in the organization. At the same time, a central authority (e.g., corporate IS department) still can establish general rules that cannot be overwritten on the workgroup level.

Two examples demonstrate the effectiveness of this strategy. Consider, for instance, employee Bill who begins routinely accessing pornographic sites on the Internet using company resources, despite the fact that such activity is prohibited by company policy. Using current technology, the company's IS department would likely not detect the activity for weeks, or even months, as the department's main focus is to keep the company's networks running smoothly, not to track individual activities. At the point when the activity is uncovered, Bill might have already violated company policy to the point where his manager has no choice but to dismiss Bill. If Bill's Internet access activity is monitored locally by Bill's supervisor, however, the supervisor can notice the prohibited activity almost immediately. After sternly reminding Bill of company policy, the supervisor can continue to monitor Bill's online activities and head off the need to terminate Bill.

As another example, consider Jane, an employee who normally has no Internet access but needs to write a competitive analysis for a new product. To complete this task effectively, Jane requires Internet access for performing required research. Conventionally, Jane's supervisor would call the company's IS department to arrange Internet access, a process requiring days or even weeks. However using the Internet access monitoring system of the present invention, Jane's supervisor can grant her access for a limited time within seconds.

3. Monitoring and regulating time spent on the Internet

Perhaps the aspect of Internet access most important to productivity of an organization is the ability to monitor and regulate the amount of time employees spend on the Internet. Although the Internet is an increasingly important business tool, it also poses a temptation to employees for use to pursue their own private interests while seemingly working.

The Internet access monitoring system of the present invention addresses this problem by allowing an organization to monitor and control the time its employees spend on the Internet. The actual monitoring can be done in a variety of ways, including monitoring the time spend by an employee to actively interact with the Internet, monitoring the total time applications access the Internet, and/or montoring the total time particular users/workstations access the Internet. The option of employing different approaches is an important one, as no one approach will likely answer all monitoring needs of an organization.

Consider, for example, the monitoring of time spent by an employee for "actively" interacting with the Internet. Active use occurs when a user directly interacts with an Internet application (e.g., Web browser) while that application accesses the Internet. This is distinguished from background use which occurs when an application or process executing in the background (i.e., does not have "focus") accesses the Internet, such as a mail client which intermittently polls an Internet-based mail server. Here, certain Internet applications, such as news collecting programs (e.g., PCN), access the Internet in the background while the user attends to other unrelated tasks in the foreground. The Internet access time of a user for a given day, therefore, is not necessarily equivalent to the time the user actively spends browsing Web sites. Conventional technologies cannot distinguish between this "active" versus "passive" access and therefore cannot present an accurate picture of the actual time the user has spent on on-line activities. Accordingly, it is desirable to monitor the time an employee spends "actively" interacting with the Internet, so that management goals of controling counterproductive Web browsing can be realized. In the currently-preferred embodiment, the system may be configured such that access to the Internet which occurs in the background is not counted against the per-day time limit imposed on users. A given application itself can be examined for determining whether it is "active" by determining whether the application receives "focus" and/or receives user input (e.g., mouse clicks or key strokes).

4. Monitoring application usage

Taken a step further, monitoring the total time particular applications access the Internet provides enhanced control. Unlike traditional monitoring technology, the Internet access monitoring system of the present invention can track Internet access on a per application basis—that is, access broken down by the application or applications used for the access. This affords much better tracking and regulating of Internet activities. By monitoring total time users/workstations access the Internet, an organization can better determine the overall Internet access requirements and loads.

The ability to monitor and regulate Internet access on a per application basis is particularly advantageous. Advantages include, for instance, the ability to specify which applications can (and cannot) access the Internet. IS departments have a strong interest in limiting the number of applications used on their LANs, including limiting available applications to a uniform set of "approved" applications. For one, user support is simplified if fewer different applications are in use. More importantly, the overall integrity of one's corporate networks is improved if known applications (or unknown versions of applications) are used. This is increasingly important as more and more applications are downloaded from the Internet, including applications which an IS department has little control over. Some of these applications might include ones which are unstable (e.g., "beta" software), have security flaws, or are even intentionally destructive (e.g., computer viruses and "Trojan Horse" programs). By monitoring abilities of individual applications to access the Internet and limiting such access to approved applications only, the Internet access monitoring system of the present invention can greatly reduce or eliminate the risk of such attacks.

In a corresponding manner, monitoring access to the Internet by individual applications allows the system of the present invention to not only track Internet traffic but also can determine in many cases data exchanged on a per application basis, including the ability to determine the name of individual files dowloaded as well as target directories to where such files are copied. The approach creates an audit trail of downloaded files, thus allowing one to trace the source of files found to contain offensive contents or pose security risks. This information can also be used to ease a user's housekeeping chores of deleting files that a Web site has copied onto the user's hard disk.

Further, per application monitoring simplifies the task of tracking bandwidth utilization for a network, including providing detailed review on how the Internet access is being used. This greatly eases planning of hardware and connection requirements. Inadvertent disruptions of the network by individual users, such as bandwidth hording by a user using RealAudio for listening to a Web audio "broadcast," can be averted.

B. Internet protocols

In order to facilitate understanding of the present invention, it is helpful to review basic architecture of the Internet and techniques for providing Internet access. For clarity, the following description of the Internet architecture focuses on those aspects which are relevant to the present invention.

The Internet is essentially an open network of computers and LANs. Computers within this open network communicate using multiple protocol layers. Each of the layers addresses a distinct concern of the communication process. As a core protocol of the Internet, Internet Protocol (IP) provides a layer for exchanging data packets between computers connected to the Internet, including providing data encapsulation and header formatting, data routing across the Internet, and fragmentation and reassembly. According to the protocol, data is transmitted by attaching a header with a destination address (IP address) and then transmitting the data packet from one computer to another until the data packet arrives at the desired destination. Along this journey, each computer uses an implementation of the IP Protocol to route the data packet to the next destination until the data packet reaches its final destination. Except for checking the integrity of the IP header, no error detection or recovery capabilities are performed. When the data packet arrives at its ultimate destination, any necessary integrity checks are carried out.

Another protocol—the transport protocol—serves as a layer responsible for guaranteeing the integrity of application data. It is, therefore, used only at the original source and final destination of the data. The Internet currently uses two different transport protocols. One protocol, User Datagram Protocol (UDP), does not offer reliable connectionless services; in practice, therefore, it is up to the target application to check data integrity. In contrast, Transmission Control Protocol (TCP), another transport portocol, provides reliable connection-oriented service, which establishes a connection with a remote computer and guarantees data integrity and delivery (or notifies the application in case of an error).

Both TCP and UDP data transmissions each provide specific headers, in addition to the IP header. In order to simplify forwarding the data packets to a target application, these headers include a port number. The port number functions to identify an application-level protocol. Port number 80, for instance, is normally used for the World Wide Web protocol (Hypertext Transport Protocol or HTTP).

TCP/IP refers to IP Protocol combined with TCP and UDP. Normally, application programs communicate with an available TCP/IP implementation (e.g., Windows "WinSock") through an Application Programming Interface (API). For Windows computers, the WinSock API simply encapsulates the TCP/IP architecture. WinSock is patterned after the popular Berkeley Sockets programming model, which is generally considered the de facto standard for TCP/IP networking.

Internet applications generally implement more specialized protocols on top of TCP/IP. For example, a Web browser implements the client portions of the HyperText Transfer Protocol (HTTP) in order to communicate with Web servers. A Web browser also might implement other protocols, such as the older File Transfer Protocol (FTP) for downloading data. Electronic mail applications (i.e., E-mail clients) implement the client portion of the Simple Mail Transfer Protocol (SMTP) and the Post Office Protocol (POP). Still other protocols exist for use in the Internet, many of which are documented in the technical, trade, and patent literature; see e.g., the Internet Engineering Task Force (IETF) RFCs ("Requests For Comments") publications available from the Internet Network Information Center (NIC), via FTP access to the NIC archive nic.ddn.mil. Due to the accelerated development of the Internet, many more protocols are unpublished or are in developmental stages. As long as a client application and a corresponding server application understand how to interpret the data packets they exchange, this generally does not pose a major problem. For applications that monitor the Internet traffic in order to detect security or other problems, however, this does pose an additional challenge. Accordingly, the preferred embodiment of the present invention is constructed to facilitate accommodation of new protocols.

Detailed Construction of the Preferred Embodiment

A. Overview

The present invention provides system and methods for client-based monitoring and filtering of access, which operates in conjunction with a centralized enforcement supervisor. In accordance with the present invention, a central filter is not employed. Instead, the present invention provides a client-side filter that is controlled by the centralized authority as long as the centralized authority has a way of enforcing non-compliance (for example by blocking access to the WAN).

At a general level, the present invention provides a system comprising one or more access management applications that set access rules for the entire LAN for one or more workgroups or individual users, a client-based filter application (installed at each client), and a central supervisor application that maintains the access rules for the client based filter and verifies the existence and proper operation of the client-based filter application. Typically, the system includes (optionally) a firewall or similar application, which works together with the supervisor application in order to block all clients that have not been verified by the supervisor application.

The access management application is employed by the LAN administrator, workgroup administrator, and/or LAN user to maintain a database of the access rules for the workstations being administrated. These access rules can include criteria such as total time a user can be connected to the Internet (e.g., per day, week, month, or the like), time a user can interactively use the Internet (e.g., per day, week, month, or the like), a list of applications or application versions that a user can or cannot use in order to access the Internet, a list of URLs (or WAN addresses) that a user application can (or cannot) access, a list of protocols or protocol components (such as Java Script™) that a user application can or cannot use, and rules to determine what events should be logged (including how long are logs to be kept). These access rules can be qualified by optionally specifying: to whom should a rule apply (list of users, list of workgroups, or all); start date and expiration date of a rule; time of day when the rule should be applied (for example from 9 am to 5 pm); whether the rule is "disclosed" to the user or workgroup manager or remains hidden; whether a rule can be overwritten or modified by the workgroup manager or user; and what should happen if a rule is violated (e.g., denying Internet access, issuing a warning, redirecting the access, creating a log entry, or the like).

The client-based filter application, which in a preferred embodiment performs all of the monitoring, logging, and filtering work, is responsible for intercepting process loading and unloading. Other responsibilities include keeping a list of currently active processes; intercepting certain keyboard, mouse and other interactive user activities in order to determine which process is actively used; intercepting and interpreting all TCP/IP communication and build a comprehensive representation of these TCP/IP activities; and intercepting certain file activity and assign them to the originating process.

By intercepting process loading and unloading and keeping a list of currently active processes, each client process can be checked for various characteristics, including checking executable names, version numbers, executable file checksums, version header details, configuration settings, and the like. With this data, the filter application can determine if the process in question should have access to the Internet and what kind of access (i.e., protocols, Internet addresses, time limitations, and the like) is permissible for the given specific user.

By intercepting and interpreting all TCP/IP communication and building a comprehensive representation of these TCP/IP activities, the system can monitor TCP/IP activities on a per process or per application basis. If a particular process has access rights to the Internet (and is permitted to use the detected protocol and no other rules are violated), the communication of the process is logged and allowed to go forward. Otherwise, the prescribed remedial action for any violated rule is performed, including logging an exception log entry and, depending on the rules the TCP/IP activity, the communication is either terminated, redirected, modified, or continued. In a similar fashion, any possible time limitation rules are evaluated and enforced at this point.

By intercepting certain file activity and assigning them to the originating process, the system can track files being created and changed by any process in order to match TCP/IP activities with corresponding file activities. If a process uses FTP to download a file, for example, the system will match that activity to a file being saved by the same process by checking file name and size. If a match is found, a log entry is generated. This allows the immediate application of internal or external virus checkers.

The centralized supervisor application is installed on a computer on the LAN that can be reached from all workstations that need access to the Internet; this is typically (although not necessarily) a server computer. The supervisor monitors whether a client has the filter application loaded and provides the filter application with the rules for the specific user or workstation. The filter application maintains a local copy of these rules so that rule enforcement continues even when the user accesses the Internet but bypasses the LAN (e.g., a mobile computer on the road). The communication between the client-based filter and the centralized supervisor application, as well as between the supervisor application and the firewall, employs encryption to ensure secure communication protocol, thus avoiding any possible attack on that level.

The system of the present invention works together with existing firewalls which allow a program (e.g., the supervisor application) to dynamically set the addresses of the workstations that should have access to the Internet. The supervisor application signals the firewall which client applications have been "certified" so that the firewall only grants Internet access to those clients. At the same time, a firewall can continue to perform its usual duties, such as protecting the LAN from outside intruders or protecting the LAN and server operating system(s).

B. System architecture

1. Terminology

For purposes of describing the architecture of the system of the present invention, it is helpful to define the following terms.

Client Monitor	The monitor component or program that runs on every workstation that can access the Internet
Client VxD	Kernel mode component of Client Monitor
Supervisor	The central program that runs on a server or a secure Client Monitor and coordinates the system
Application	Third party application that can access the Internet or WAN via the WinSock API or a similar API
Firewall	Third party filter program that sits between the LAN and the Internet
Host	Third party server prograrn that can be contacted through the Intemet
ISP Server	Internet Service Provider Server application that authenticates user and serves as gateway to Internet
ISP Supervisor	Version Qf Supervisor that coordinates Internet access with ISP Server
RAS	Remote Access Service component of Windows 95/NT that dials the remote computer and initializes the contact
ISP Authentication Server	Internet Service Provider Server application that authenticates user and serves as gateway to Internet
ISP "Sandbox" Server	HTTP Server used when client has only restricted Internet access
ISP POP	Internet Service Provider Point-Of-Presence comprising modems, server, and router
ISP POP Server	Server component of ISP POP

2. LAN-based embodiment

FIG. **3A** provides an overview of an Internet-based (client/server) system **300** in which the present invention may be embodied. As shown, the system includes multiple clients **310** (e.g., clients **310***a*, **310***b*, **310***c*, each of which comprises a personal computer or workstation, such as system **100**) connected to a network **320**, such as a Windows NT Local Area Network (Microsoft Corporation of Redmond, Wash.). Each client includes a client-side monitoring component for monitoring Internet access in accordance with the present invention, as specifically shown at **311***a*, **311***b*, and **311***c*. The network **320** is connected to a server **321** (or another client) having a supervisor or verifier component **323**. The supervisor component **323** provides independent verification of the clients, for allowing or disallowing requests of each particular client. In effect, the supervisor **323** directs runtime monitoring operations.

The network **320** itself can be a server-based network (e.g., Windows NT Server providing services to network clients) or, alternatively, a peer-to-peer network. Communications to the outside (e.g., Internet) are typically achieved using TCP/IP protocol. The local network **320** communicates with the Internet, shown at **340**, preferably through a "firewall" **330**. The firewall **330** itself may be implemented in a conventional manner, such as employing a router-based or server-based firewall process for monitoring communications with various Web servers **350** connected to the Internet **340**.

With reference to FIG. **4**, client-side operation of the system is shown in further detail. As shown in FIG. **4** for client **410**, a given client generally includes one or more applications (e.g., applications **421**, **423**) which require Internet access. A Web browser (e.g., Netscape Navigator™ or Microsoft Internet Explorer™ browser software) is but one of a multitude of such applications. Each application, in turn, communicates directly with a client-side communication driver, such as Winsock driver **430**—a Windows implementation and encapsulation of TCP/IP.

The client **410** includes a client-side monitor—data acquistion module **440**—which "hooks into" the communication driver **430**. In the instance of Windows Winsock communication driver, for example, a process can hook into the driver using Winsock VxD extensions. As the various applications submit requests to the communication driver **430**, the data acquisition module **440** can intercept the communications for determining whether the request is permitted under the rules. For instance, when a request for access is received from an application, the monitor first verifies that, according to the rules in place, such an application is permitted to access the Internet. Rules currently in force might specify that only particular applications (or particular versions of those applications) can access the Internet; all other applications are denied access. For a Winsock-based implementation, the data acquisition module **440** can, in effect, trap the request at the VxD driver level, thereby effectively blocking the request at the level of the Winsock communication driver.

In addition to checking whether the application itself should have access, the data acquisition module **440** monitors the individual messages which are exchanged between the applications and the communication driver. For instance, the data acquisition module **440** can trap an HTTP "SEND" command generated by an application. By analyzing the message and any accompanying context information, the data acquisition module **440** can determine whether the command is permitted under the rules in place. By examining port address context information provided with the command, for example, the data acquisition module **440** can readily determine the underlying protocol. For Web server access, the data acquisition module **440** would identify HTTP—the underlying protocol used to communicate with the Web. Having determined the protocol, the data acquisition module **440** would verify that the protocol is permitted for the application (or for the client). As an example of a typical rule which might be in place, a supervisor might establish a rule blocking FTP (file transfer protocol) by Web browsers, for preventing users from tying up the network with large FTP file transfers.

The actual flow of messages or message "traffic" monitored by the system is perhaps best illustrated by example. An exemplary use of the system for accessing a Web site generates the following trace of messages.

```
#0000   Msg:    10020024  Process:   ffff38ff      Handle:  c307bd1c  open
        Len:    00000016  Address:   82859080
        C:\WINDOWS\SYSTEM.DAT
#0001   Msg:    0002000b  Process:   ffff38ff      Handle:  c307bd1c  close
        Size: 00000000:000a752c
#0002   Msg:    00010010  Process:   fffae7b7      Handle:  00000000  socket
#0003   Msg:    00010110  Process:   fffae7b7      Handle:  c34075cc  socket-x
#0004   Msg:    80010003  Process:   fffae7b7      Handle:  c34075cc  connect
        Family: 0002 Port: 0053 IP: 204.94.129.65
#0005   Msg:    00010103  Process:   fffae7b7      Handle:  c34075cc  connect-x
        Family: 0002 Port: 0053 IP: 204.94.129.65
#0006   Msg:    00010004  Process:   fffae7b7      Handle:  c34075cc  getpeername
#0007   Msg:    00010104  Process:   fffae7b7      Handle:  c34075cc  getpeername-x
#0008   Msg:    0001000d  Process:   fffae7b7      Handle:  c34075cc  send
        Family: 0002 Port: 0053 IP: 204.94.129.65
         Len:   00000023  Address:   82859890
     00:
#0009   Msg:    0001010d  Process:   fffae7b7      Handle:  c34075cc  send-x
#000a   Msg:    0001000a  Process:   fffae7b7      Handle:  00000000  select_setup
#000b   Msg:    0001010a  Process:   fffae7b7      Handle:  00000000  select_setup-x
#000c   Msg:    0001000b  Process:   fffae7b7      Handle:  00000000  select_cleanup
#000d   Msg:    0001010b  Process:   fffae7b7      Handle:  00000000  select_cleanup-x
#000e   Msg:    80010003  Process:   fffae7b7      Handle:  c34075cc  connect
#000f   Msg:    00010103  Process:   fffae7b7      Handle:  c34075cc  connect-x
#0010   Msg:    00010004  Process:   fffae7b7      Handle:  c34075cc  getpeername
#0011   Msg:    00010104  Process:   fffae7b7      Handle:  c34075cc  getpeername-x
        Result: 00002749 WSAENOTCONN
#0012   Msg:    0001000d  Process:   fffae7b7      Handle:  c34075cc  send
        Family: 0002 Port: 0053 JP: 204.94.129.66
         Len:   00000023  Address:   8285a290
     00:
#0013   Msg:    0001010d  Process:   fffae7b7      Handle:  c3407Scc  send-x
        Family: 0002 Port: 0053 IP: 204.94.129.66
#0014   Msg:    0001000a  Process:   fffae7b7      Handle:  00000000  select_setup
#0015   Msg:    0001010a  Process:   fffae7b7      Handle:  00000000  select_setup-x
#0016   Msg:    0001000b  Process:   fffae7b7      Handle:  00000000  select_cleanup
#0017   Msg:    0001010b  Process:   fffae7b7      Handle:  00000000  select_cleanup-x
#0018   Msg:    00010009  Process:   fffae7b7      Handle:  c34075cc  recv
```

-continued

```
#0019   Msg:     00010109  Process:   fffae7b7     Handle:  c34075cc   recv-x
        Len:     000000ad  Address:   8285a990
      00: 00 06  85  80  00  01  00  03  00  02  00  02  03  77  77  77  . . . . . . . . . . . . .www
      10: 09 77  65  62  6d  6f  6e  6b  65  79  03  63  6f  6d  00  00  .webmonkey.com. .
      20: 01 00  01  c0  0c  00  01  00  01  00  00  a8  c0  00  04  cc  . . . . . . . . . . . . . . . .
      30: 3e 81  13  c0  0c  00  01  00  01  00  00  a8  c0  00  04  cc  >. . . . . . . . . . . . . . .
      40: 3e 81  93  c0  0c  00  01  00  01  00  00  a8  c0  00  04  cc  >. . . . . . . . . . . . . . .
      50: 3e 83  93  09  77  65  62  6d  6f  6e  6b  65  79  03  63  6f  >. . .webmonkey.co
      60: 6d 00  00  02  00  01  00  00  a8  c0  00  0f  03  6e  73  31  m . . . . . . . . . . . .ns1
      70: 08 68  6f  74  77  69  72  65  64  c0  5d  c0  53  00  02  00  .hotwired.].S . . .
      80: 01 00  00  a8  c0  00  06  03  6e  73  32  c0  70  c0  6c  00  . . . . . . . .ns2.p.1.
      90: 01 00  01  00  00  a8  c0  00  04  cc  3e  84  20  c0  87  00  . . . . . . . . . .>. . .
      a0: 01 00  01  00  00  a8  c0  00  04  cc  3e  82  7c              . . . . . . . . . . .>.|
#001a   Msg:     00010002  Process:   fffae7b7     Handle:  c34075cc   closesocket
#001b   Msg:     00010102  Process:   fffae7b7     Handle:  c34075cc   closesocket-x
#00lc   Msg:     00010010  Process:   fffae7b7     Handle:  000000CO  socket
#001d   Msg:     00010110  Process:   fffae7b7     Handle:  c34075cc   socket-x
#001c   Msg:     00010007  Proccss:   fffac7b7     Handlc:  c34075cc   ioctisockct
#001f   Msg:     00010107  Process:   fffae7b7     Handle:  c34075cc   ioctlsocket-x
#0020   Msg:     00010001  Process:   fffae7b7     Handle:  c34075cc   bind
#0021   Msg:     0O010101  Process:   fffae7b7     Handle:  c34075cc   bind-x
#0022   Msg:     80010003  Process:   fffae7b7     Handle:  c34075cc   connect
        Family: 0002 Port: 0080 IP: 204.62.129.147
#0023   Msg:     00010103  Process:   fffae7b7     Handle:  c34075cc   connect-x
        Result: 00002733 WSAEWOULDBLOCK
        Family: 0002 Port: 0080 IP: 204.62.129.147
#0024   Msg:     0001000a  Process:   fffae7b7     Handle:  00000000   select_setup
#0025   Msg:     0001010a  Process:   fffae7b7     Handle:  00000000   select_setup-x
#0026   Msg:     0001000b  Process:   fffae7b7     Handle:  0000000a   select_cleanup
#0027   Msg:     0001010b  Process:   fffae7b7     Handle:  00000000   select_cleanup-x
#0028   Msg:     00010009  Process:   fffae7b7     Handle:  c0f005f4   recv
#0029   Msg:     00010109  Process:   fffae7b7     Handle:  c0f00Sf4   recv-x
        Family: 0002 Port: 1028 IP: 127.0.0.1
         Len:    00000001  Address:   8285b990
      00: @
#002a   Msg:     0001000a  Proccss:   fffac7b7     Handlc:  00000000   sclcct_sctup
#002b   Msg:     0001010a  Process:   fffae7b7     IIandle: 00000000   select_setup-x
#002c   Msg:     0001000b  Process:   fffae7b7     Handle:  00000000   seiect_cleanup
#002d   Msg:     0001010b  Process:   fffae7b7     Handle:  000C0000   select_cleanup-x
#002e   Msg:     0001000e  Process:   fffae7b7     Handle:  c34075cc   setsockopt
#002f   Msg:     0001010e  Process:   fffae7b7     Handle:  c34075cc   setsockopt-x
#0030   Msg:     0001000e  Process:   fffae7b7     Handle:  c34075cc   setsockopt
#0031   Msg:     0001010e  Process:   fffae7b7     Handle:  c3407Scc   setsockopt-x
#0032   Msg:     0001000e  Process:   fffae7b7     Handle:  c34075cc   setsockopt
#0033   Msg:     0001010e  Process:   fffae7b7     Handle:  c34075cc   setsockopt-x
#0034   Msg:     00010004  Process:   fffae7b7     Handle:  c34075cc   getpeername
#0035   Msg:     00010104  Process:   fffae7b7     Handle:  c34075cc   getpeername-x
#0036   Msg:     8801000d  Process:   fffae7b7     Handle:  c34075cc   send
        Family: 0002 Port: 0080 IP: 204.62.129.147
        Len:     0000011d  Address:   c279a3a0
      00: GET / HTTP/1.0
      10: Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
      52: Accept-Language: en
      67: UA-pixels: 640 × 480
      7b: UA-color: color8
      8d: UA-OS: Windows 95
      a0: UA-CPU: ×86
      ad: User-Agent: Mozilla/2.0 (compatible; MSIE 3.01; Windows 95)
      ea: Host: www.webmonkey.com
      03: Connection: Keep-Alive
      1b:
#0037   Msg:     0001010d  Process:   fffae7b7     Handle:  c34075cc   send-x
#0038   Msg:     00010009  Process:   fffae7b7     Handle:  c34075cc   recv
#0039   Msg:     00010109  Process:   fffae7b7     Handle:  c34075cc   recv-x
        Result: 00002733 WSAEWOULDBLOCK
        Family: 0002 Port: 0080 IP: 204.62.129.147
#003a   Msg:     0001000a  Process:   fffae7b7     IIandle: 00000000   select_setup
#003b   Msg:     0001010a  Process:   fffae7b7     Handle:  00000000   select_setup-x
#003c   Msg:     0001000b  Prccess:   fffae7b7     Handle:  00000000   select_cleanup
#003d   Msg:     0001010b  Process:   fffae7b7     Handle:  00000000   select_cleanup-x
#003e   Msg:     00010009  Process:   fffae7b7     Handle:  c34075cc   recv
#003f   Msg:     88010109  Process:   fffae7b7     Handle:  c34075cc   recv-x
        Family: 0002 Port: 0080 IP: 204.62.129.147
        Len:     00000149  Address:   c279a544
      00: HTTP/1.0 302 Found
      14: Date: Fri, 03 Jan 1997 23:02:57 GMT
      39: Server: Apache/1.1.1 HotWired/1.0
      5c: Location: http://www.webmonkey.com/webmonkey/
      8b: Content-type: text/html
```

-continued

```
        a4:
        a6: <HEAD><TITLE>Document moved</TITLE></HEAD>
        d1: <BODY><H1>Document moved</H1>
        ef: The document has moved <A
HREF="http://www.webmonkey.com/webmonkey/">here
        38: </A>.<P>
        41: </BODY>
#0040   Msg:    00010009  Process:   fffae7b7    Handle:  c34075cc   recv
#0041   Msg:    00010109  Process:   fffae7b7    Handle:  c34075cc   recv-x
        Family: 0002 Port: 0080 IP: 204.62.129.147
#0042   Msg:    00010002  Process:   fffae7b7    Handle:  c34075cc   closesocket
#0043   Msg:    00010102  Process:   fffae7b7    Handle:  c34075cc   closesocket-x
#0044   Msg:    00010010  Process:   fffae7b7    Handle:  00000000   socket
#0045   Msg:    00010110  Process:   fffae7b7    Handle:  c34075cc   socket-x
#0046   Msg:    00010007  Process:   fffae7b7    Handle:  c34075cc   ioctlsocket
#0047   Msg:    00010107  Process:   fffae7b7    Handle:  c3407Scc   ioctlsocket-x
#0048   Msg:    00010001  Process:   fffae7b7    Handle:  c34075cc   bind
#0049   Msg:    00010101  Process:   fffae7b7    Handle:  c34075cc   bind-x
#004a   Msg:    80010003  Process:   fffae7b7    Handle:  c34075cc   connect
        Family: 0002 Port: 0080 IP: 204.62.131.147
#004b   Msg:    00010103  Process:   fffae7b7    Handle:  c34075cc   connect-x
        Result: 00002733 WSAEWOULDBLOCK
        Family: 0002 Port: 0080 IP: 204.62.131.147
#004c   Msg:    0001000a  Process:   fffae7b7    Handle:  00000000   select_setup
#004d   Msg:    0001010a  Process:   fffae7b7    Handle:  00000000   select_setup-x
#004e   Msg:    0001000b  Process:   fffae7b7    Handle:  00000000   select_cleanup
#004f   Msg:    0001010b  Process:   fffae7b7    Handle:  00000000   select_cleanup-x
#0050   Msg:    0001000e  Process:   fffae7b7    Handle:  c34075cc   setsockopt
#0051   Msg:    0001010e  Process:   fffae7b7    Handle:  c34075cc   setsockopt-x
#0052   Msg:    0001000e  Process:   fffae7b7    Handle:  c34075cc   setsockopt
#0053   Msg:    0001010e  Process:   fffae7b7    Handle:  c34075cc   setsockopt-x
#0054   Msg:    0001000e  Process:   fffae7b7    Handle:  c34075cc   setsockopt
#0055   Msg:    0001010e  Process:   fffae7b7    Handle:  c34075cc   setsockopt-x
#0056   Msg:    00010004  Process:   fffae7b7    Handle:  c34075cc   getpeername
#0057   Msg:    00010104  Process:   fffae7b7    Handle:  c34075cc   getpeername-x
#0058   Msg:    8801000d  Process:   fffae7b7    Handle:  c34075cc   send
        Family: 0002 Port: 0080 IP: 204.62.131.147
        Len:    00000127  Address:   c279a3a0
        00: GET /webmonkey/ HTTP/1.0
        1a: Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
        5c: Accept-Language: en
        71: UA-pixels: 640 × 480
        85: UA-color: color8
        97: UA-OS: Windows 95
        aa: UA-CPU: ×86
        b7: User-Agent: Mozilla/2.0 (compatible; MSIE 3.01; Windows 95)
        f4: Host: www.webmonkey.com
        0d: Connection: Keep-Alive
        25:
```

As shown, the trace shows commands for "file open" and "file close." This is followed by a command for creating a socket and a command for connecting to a particular site. At this point, for instance, one can discern that the system is using port **53**—the DNS port—for looking up an address (IP address) on the Web. In response to this request, the system receives a series of IP addresses, which have been encoded in a particular format. Once the IP address has been received by the system, the particular Internet application making the request can now issue various commands (e.g., HTTP "GET" command) for retrieving information from a particular Web site. In response to these requests, the corresponding server at the Web site sends appropriate responses, including transmitting requested content. Since the system of the present invention monitors the message traffic at the level of individual messages, the system is able to selectively block access, as dictated by the configurable rules.

For determining whether the requested access of the client's application is accessing a permitted site, the data acquisition module **440** examines the IP address for the site which the application seeks to communicate with and compares that address against a list of allowed addresses (or conversely against a list of disallowed addresses). Certain sites can have multiple IP addresses. Accordingly, the system of the present invention stores the IP addresses with the respective Web sites, so that a particular site can be resolved at the level of its individual IP addresses. In this manner, the system of the present invention permits access based either on a Web site name (e.g., www.cnn.com) or based on a particular IP address.

As shown in FIG. **5**, the monitor or data acquisition module **440**, which is preferably implemented as a Windows VxD driver, includes the following subcomponents. Winsock Hook **501** includes functionality for connecting or "hooking" into the Winsock communication driver. File Hook **503**, in a similar manner, includes functionality allowing the driver **440** to hook into the file subsystem provided by the underlying operating system. Process Hook **505**, another subcomponent which hooks into the underlying operating system, tracks all currently-executing applications or processes. All of these hooks, as implemented in a Windows VxD, are capable of executing at ring **0**—that is, execution at the highest privileged level of the operating system.

Also shown, the module **440** includes a buffer interface (manager) **507** which interfaces directly with a client system-maintained FIFO (first-in, first-out) message buffer **550**. The message buffer **550** comprises an array of messages maintained in a fixed-size (i.e., pre-allocated) block of client memory. The buffer **550** itself is shared between the data acquisition module **440** and the various other executing applications. In a preferred embodiment, the data acquisition module **440** utilizes the shared message buffer **550**, so that the module itself need not undertake various allocation/deallocation operations (which might degrade performance).

Actual access to individual messages within the buffer is achieved by mapping globally-accessible selectors, thus permitting the data acquisition module **440** with the ability to access individual messages (and content thereof) within the message buffer **550**. With direct access to the underlying messages themselves, the data acquisition module can patch (i.e., modify) dynamically at runtime the content of various messages. For instance, a request to access a particular Web site can be patched to instead redirect that request to another site. More generally, since the module **440** can trap individual messages, individual messages can be modified in an arbitrary manner according to the rules in place, including disallowing (i.e., blocking) specific messages.

Actual interpretation of individual messages is performed by data interpretation module **560** which co-exists with the data acquisition module **440**. The data interpretation module **560**, which communicates directly with the data acquisition module **440**, keeps track of all currently-executing processes. To intelligently determine the action which it should undertake for a given message, the module **560** refers to a rules database or knowledgebase **570**. Here, the various rules which define permitted activity in the Internet-based system are stored in a format which is readily accessible by the data interpretation module **560**. Contained within the rules database **570** are individual databases fully characterizing the administrator-specified rules for the system. When the data interpretation module **560** is first loaded at a given client machine, it attempts to download from the supervisor module into its local knowledgebase **570** a copy of those system rules pertinent to the client. In the event that such a copy is not available (or has not changed since last download), the data interpretation module **560** employs the last downloaded local copy.

Finally, the data interpretation module **560** maintains log information, including an audit (transaction) log **580** and an exception log **585**. The former provides a time-sequence log of messages processed by the system; the latter provides a time-sequence log of exceptions (e.g., access violations attempted by users) which have occurred in the system.

3. ISP-based embodiment

Although the previously-described embodiment focuses on monitoring LAN-based access to the Internet and other WANs, the present invention can alternately be implemented for establishing a monitoring and filtering system for Internet Service Providers (ISPs) or similar organizations. This allows ISPs to offer their users a tamper-proof, safe, and managed access to the Internet and protects the users from many security threats. It enables users to control how, when, and who accesses the Internet with their account.

FIG. **3B** illustrates certain modifications to the system **300** (of FIG. **3A**) for creating the alternative embodiment, shown as system **300***a*. Most ISPs use decentralized installations called Points-Of-Presence (POPs), such as POP **320***a*. These installations comprises a series of modems to connect to client PCs or client LANs, a server or LAN, and one or more router to connect the installation to the Internet (normally via high-speed dedicated lines). ISPs normally have one or more POPs in the areas that they serve. When a user dials into a POP (e.g., using a protocol such as SLIP), the POP server in return contacts the central ISP authentication server either via the Internet or a dedicated line. The central authentication server checks the user's ID and password and signals the POP server whether the user is allowed or denied access to the Internet. If the user is allowed to access the Internet, the POP enables the access autonomously without further involvement of central ISP servers.

In this alternate embodiment, the ISP installs an additional central server component **370** to host the central supervisor application; this new component comprises an ISP authentication server **371** and an ISP supervisor server **372** (which includes a central supervisor application **373**). After the central ISP authentication server **371** has established the authenticity of the user, it contacts the central supervisor application **373** in order to find out if the user has established additional access monitoring services. In such a case, the ISP authentication server **371** signals the POP server **320***a* to only allow limited access to the Internet and redirect all requests to a "Sandbox" server application, shown at **374**, on the central supervisor server **372**. This "Sandbox" server **374** restricts the client's Internet access to a very limited account maintenance site.

The Client Monitor on the client PC (e.g., monitor **311***a*) monitors the log-on process. Once the limited access to the Internet is established, the monitor contacts the central supervisor application **373** on the ISP supervisor server **372** in order to receive access rules and other required components. Once the central supervisor application **373** is satisfied that the Client Monitor has received the appropriate access rules and is working satisfactory, it contacts the POP server **320***a* to signal that the user now has full Internet access. The central supervisor application **373** will continue to check the Client Monitor and, in case of any problems, signals the POP server **320***a* to fall back to limited access to the Internet.

If the user does not have a Client Monitor installed, or the Client Monitor is not functioning or has been tampered with, the user will only have access to the "Sandbox" server **374**. The user will not gain access to the rest of the Internet until the user downloads the Client Monitor component from the "Sandbox" server **374** or otherwise reinstall the Client Monitor application.

C. Preferred user interface

1. General

The client-side monitoring component provides a preferred user interface **600**, as shown in FIG. **6A**. The interface **600** serves to display the user's current Internet activity and/or past log. As illustrated, the interface **600** includes a main menu **601**, a selection or tool bar **605**, a Web applications panel **610**, a contents panel **620**, and a details panel **630**. The tool bar **605** provides a display filtering mechanism, affecting the actual information displayed by the various panels. For instance, the user can employ the tool bar **605** for selecting what type of information to show (e.g., applications), which user the system should display information for (e.g., the current user or another named user), and what time frame is of interest to the user (e.g., "today"). Selection icons **640**, positioned along one side of the interface **600**, provide one-click access to user commands (which correspond to those available from the menu **601**).

FIG. **6B** illustrates appearance of the interface **600** (now **600***a*) during operation of a Web browser (e.g., Netscape Navigator™ or Microsoft Internet Explorer™ browser software). The applications panel **610** (now **610***a*) shows the currently-executing applications or processes. As shown at **611**, current Web processes for this example include Internet Explorer. In the currently-preferred embodiment, processes are illustrated in an outline (hierarchical) view, with individual processes represented by nodes of the outline. Upon the user selecting to expand an application node (e.g., by clicking on node **611**), the system, in response, displays dependent or child nodes representing protocols employed by that application. For the application node **611**, for instance, the system displays child nodes **612**.

In a corresponding manner, the contents panel **620** and the details panel **630** provide further information for the currently-selected application or process. Specifically, the contents panel **620** (now shown as **620***a*) displays the details (contents) which comprise the Web object (e.g., Web page) selected by the user (e.g., from a Web browser). In conjunction with this contents view, the details panel **630** (now shown at **630***a*) displays a details or transaction list—a list of transactions (e.g., HTTP commands) which have occurred for the currently-selected Web object. In the example shown in FIG. **6**B, for instance, the display transactions include HTTP "GET" commands, for getting (fetching) various bitmaps which comprise the Web page currently viewed in the browser. FIG. **6**C illustrates appearance of the interface **600** (now **600***c*) as additional Web processes are launched. For instance, the applications panel **610** (now **610***c*) displays a new node **613**, for indicating the new executing process, here WebFerret (a search utility application).

Using the current example of a system executing Internet Explorer and WebFerret, operation of the interface **600** for monitoring protocols will now be illustrated. FIG. **6**D illustrates the interface **600** (now **600***d*) with full expansion of the "DNS" protocol node **650**, as shown in the applications panel **610** (now **610***d*). The DNS or Directory Name Service protocol is the main protocol employed to look up the address of a name (i.e., "www.cnn.com") on the Internet. For the current example, the user has "visited" two Web sites: Starfish Software (www.starfishsoftware.com) and CNN (www.cnn.com). These are illustrated as children nodes **651**, which depend from (i.e., hang off of) the DNS node **650**. Upon the user selecting a particular one of the dependent nodes **651**, the interface **600***d* displays corresponding information in the contents panel **620***d*. For the particular example of a DNS-based Web name of www.cnn.com, the contents panel **620***d* displays corresponding Internet addresses, as shown at **653**. In this specific example, the CNN Web site has five Internet or IP (Internet protocol) addresses.

During system operation, therefore, the Internet monitor performs monitoring of different types of Internet access protocols. Support for any given type of access protocol is provided via a dynamically-loaded driver. For instance, the system includes driver supporting HTTP, FTP, SMTP, and POP3 protocols. Other drivers are provided for different content types, for instance, supporting parsing of HTTP files, executable files, ZIP files, ActiveX controls, or Java classes. Each driver is responsible for monitoring and filtering access for its particular type, including ensuring that any user activity which employs that access type conforms to any rules or conditions specified for the Internet monitor.

FIG. **6**E illustrates operation of the interface **600** (now **600***e*) for the WebFerret application. As shown at **655**, the application employs the DNS protocol for looking up the addresses of several Web search engines, including Alta Vista™, Yahoo™, Infoseek™, and Lycos™ search engines.

In a corresponding manner, as indicated at **657**, the WebFerret application employs World Wide Web protocol (HTTP) for communicating with each of the foregoing Web search engines.

2. Rule wizard interface

The system allows user (e.g., administrator) configuration of rules which govern use of the protocols monitored by the

system. For instance, an administrator can establish a rule based on a particular application, such as a rule presenting Internet access by a real audio player application (ra32.exe). Rules can also be established on the basis of including and/or excluding access to particular Internet sites. For instance, an administrator can establish a rule allowing users to only access a limited number of approved sites. On the other hand, the administrator can set a rule blocking user access to particular sites (e.g., pornographic sites). Rules can also be set which are time-based in nature. For instance, an administrator can establish a rule setting a time limit (e.g., 30 minutes) for how long a user can access the Internet each day. More important in the business environment, the system allows such a time limit to be set according to a user's "active" use of the Internet.

FIGS. 7A–K illustrate a preferred user interface or "wizard" dialogs for configuring rules. As shown in FIG. 7A, a preferred interface **700** provides a "view" of rules governing operation of the Internet access monitor, displaying all of the rules which are available for a current configuration. As shown, the interface **700** includes a toolbar **710** having the following buttons. Button **711** allows the user to create a new rule. Button **712** allows the user to edit an existing rule. Button **713** allows the user to delete a rule. Button **714** allows the user to print out a hard copy of a rule. Finally, button **715** allows the user to group rules together (i.e., into user-defined groups). The toolbar **710** also includes a group list field **716** and a user list field **717**. Together, these fields allow the user to manage groups of computers and/or groups of users.

Below the toolbar **710** is a rule panel **721** which lists the current rules in place for the system (for the currently-selected group). For each rule, the rule panel **721** displays a description, a start date, and an expiration date (if any). In this fashion, individual rules can be presented in a self-explanatory manner. For instance, rule **723** specifies that Web browsing is restricted to one hour per day for weekdays, from 9 a.m. to 6 p.m. The rule, which has a start day of Sep. 12, 1996, is currently configured to never expire. The rule panel **721** displays all rules, whether they are created by the system administrator or a group manager, or pre-existing (default) rules provided by the system. Enforcement of any given rule can be suspended by "disabling" the rule, such as shown at **724**.

Below the rule panel **721**, the interface **700** includes a rule details panel **730**. As illustrated by tabs **731**, the panel **730** itself includes the following pages: general, history, and comment. The details panel **730** provides detail information for the particular rule which is currently selected in the panel **721**. For the rule **723** as the currently selected rule, for instance, the panel **730** displays on its "general" page detail information which describes general features of the rule. On the "history" page (not shown), the panel **730** displays history/revision information for the selected rule. In a similar manner, on the "comments" page (not shown), the panel **730** provides any comments which the user has entered for the rule. The interface **700** also includes a status bar **735** which displays, at **736**, a rule ID (internally-maintained identifier) for the currently-selected rule. The ID is employed internally for tracking and cross referencing rules. The status bar **735** also displays current on-line activity at **737**, such as a status message indicating which objects are currently being received from a Web site.

Operation of the interface **700** will now be described by demonstrating the user task of creating a new rule. The user begins the process by selecting "new rule" button **711**. In response, the system displays rules expert or wizard dialog

740, as shown in FIG. **7**B. At the outset, the wizard dialog **740** asks the user what kind or type of new rule should be created. Generally, any given rule is a combination of access rights granted on the basis of available applications, permitted time limits, permitted user activities, permitted protocols, and the like. At the outset, the system provides pre-defined or "canned" rules which pre-package access rights (based on common combinations of the foregoing attributes). The wizard dialog **740** provides a list **741** of rule types currently defined for the system. As shown, for instance, the user can select type **742** for limiting what applications (including individual applications) can do on the Internet.

After selecting a rule type, the user proceeds to the next pane or page of the wizard dialog (by clicking the "next" button). As shown in FIG. 7C, the wizard dialog **740** (now **740***a*) displays a "title" edit field **743** and a "comments" edit field **744**. Here, the user can enter information for the title and comments, respectively.

Proceeding to the next page, the wizard dialog **740** (now **740***b*) displays an outline list **745** of all applications known to the system. Using include/exclude buttons **746**, the user can instruct the system to selectively include or exclude applications (and versions thereof) which are to be affected by the new rule. The user's current selection is displayed at application list **747**. The applications affected by a given list are the cumulative sum of the applications on the list. For instance, list **747** indicates that all applications are added to the list except for Internet Explorer and Netscape Navigator (as these have been excluded from the list). If desired, the user can save a particular list as a user-defined group. For the list shown at **747**, for example, the user could save the list as "All applications except Web browsers."

Note also that the applications outline **745** provides a version-based list of applications. Under "Internet Explorer," for instance, the outline displays "latest release," "version 3.02," and subsequent prior versions (not shown). By tracking applications on a per version basis, the system can selectively enforce rules against individual versions of an application. Versions 3.01 and 3.00 of Microsoft's Internet Explorer™browser software, for instance, have known security flaws. Using a per-version rule in accordance with the present invention, the user can create a rule blocking Internet access for Internet Explorer versions 3.01 and 3.00, yet allow access for other versions (e.g., version 3.02).

Internally, the system defines "latest release" and "all versions" for each application. "Latest release" represents the most recent version of a given application. As each new version of an application is released, "latest release" is automatically updated to that most recent release. "All versions" is defined internally, in contrast, to simply include all versions of a given application.

After specifying which application or applications (and versions thereof) should be affected by the new rule, the user proceeds to specify which activities are to be limited. As illustrated in FIG. 7E, the wizard dialog **740** (now **740***c*) includes an activity pane which allows the user to select one or more activities limited by the new rule. In a manner similar to that described for selecting applications, activities are selected from an outline list **755** for inclusion or exclusion, by using include/exclude buttons **756**. Again, the user creates a set representing the sum of included or excluded activities; these are displayed by outline list **757**. In an exemplary embodiment, the system provides default activities which can be limited, including, for instance, Worldwide Web (Internet) activity, receiving incoming e-mail, and sending outgoing e-mail. The limitation on outgoing e-mail can be employed, for example, to prevent unknown "spy" applications from using e-mail services (e.g., Microsoft MAPI) to steal confidential information from the user's system. Note that a firewall, in contrast, cannot provide an effective defense against such spy applications, because firewalls do not have the capability to understand the underlying applications. For the new rule being created for the example at hand, all activities have been restricted for all applications except Internet Explorer™ browser software (all versions) and Netscape Navigator™ browser software (all versions).

Also shown, the wizard dialog **740***c* includes an "advance" button **777** which allows the user to define his or her own activities. The user-defined activities are defined based on Internet port, address families, and the like. In this manner, the user can create his or her own Internet access activities for use in rules.

The user is now ready to specify to which people and/or to which computers the new rule is to apply. As shown in FIG. 7F, the wizard dialog **740** (now **740***d*) includes a pane which allows the user to define a set which includes or excludes people, computers, and/or groups thereof. In a manner to that previously described for defining activities and for specifying applications, the pane includes an outline list **761** from which the user can select to include or exclude items. People, computers, and groups which have been included or excluded are displayed on the right hand side of the dialog, by list **763**. As before, the list is a set representing the cumulative sum of items which have been included or excluded.

"People" represent individual users who can log on to the system (from one or more computers). A "computer", on the other hand, represents an individual workstation or other device connected to the system; typically, such a device has a unique IP address assigned to it. A "group" represents a set which includes or excludes certain people and/or computers. This approach permits the system to allow a Web server (a device), for instance, to have unlimited Internet access regardless of which user is logged onto that computer. At the same time, the system can prevent a given user from undertaking certain activity, regardless of which computer that user has logged onto. By using groups, the user can conveniently encapsulate certain people or computers (or subgroups thereof) into a user-specified group, such as a "marketing" group. For the new rule of the example at hand, the rule disallows Internet access for all applications except Internet Explorer™ and Netscape Navigator™ browser software for all users and computers except for the marketing group, the Web server computer, and one individual (the user having username of gfreund).

FIG. **7**G illustrates the next pane of the wizard dialog **740** (now **740***e*). Here, the wizard dialog **740** displays a selection of actions **767** which the system should undertake in the event of an attempted rule violation. Choices include, for instance, stopping the activity and showing an error dialog, stopping the activity and redirecting the user to an error page (when possible), stopping the activity and generating an application error, and the like. Additionally, the user can specify at this point that the system should generate an entry in a system alert or error log. This last option is helpful, for instance, when the access monitor is first deployed; this allows the system administrator to establish a log of potential rule violations before he or she decides to activate rule enforcement. For those response actions associated with a dialog, the wizard dialog provides a dialog text field **769** for entering messages. Messages can include tags or macros for allowing certain text to be specified at runtime, such as replacing <application name> or <Web site name> with the current application name or Web site name, respectively, for the instant rule being violated.

As illustrated in FIG. **7**H, the wizard dialog **740** (now **740***f*) includes a start date/expiration date pane. Here, the user can specify the start date and expiration date (if any) for the new rule being created. Further, the user can also specify particular time intervals (e.g., during weekdays and/or during weekends) when the rule is enforced. This allows an administrator, for instance, to specify that a rule blocking a RealAudio application remains in force during working hours on weekdays—that is, at times when network traffic is already congested. At other times, however, the rule is not enforced. For the example shown in FIG. **7**H, the rule has a start date of Mar. 31, 1997 and never expires; the rule is enforced weekdays and weekends from 8 a.m. to 5:30 p.m.

As shown in FIG. **7**I, the wizard dialog **740** (now **740***g*) includes a pane allowing the user to specify "who," if anybody, can modify or suspend the new rule. Recall that, in general, rules will be modified by a system administrator or by workgroup or department supervisors. Additionally, however, an end user might be in a position to modify or suspend rules, particularly for those rules which the end user has created to control Internet access by applications on his or her system. In this manner, the system of the present invention allows rules to be modified in a distributed manner. Thus, an organization can control network use (through Internet access) along organizational or personnel structures, without requiring involvement of IS personnel. Accordingly, the dialog **740***g* provides check boxes **771** permitting the new rule to be modified or suspended by: (1) a workgroup supervisor, (2) a department supervisor, and/or (3) end users. For the present example, since none of the check boxes have been "checked," the new rule cannot be modified by anyone other than the user who has authority to create new system-wide rules (e.g., system administrator).

As illustrated in FIG. **7**J, the wizard dialog **740** (now **740***h*) next displays a summary pane allowing the user to review the new rule. In particular, the dialog **740***h* displays a rule summary **775** comprising a prose description summarizing user input for the wizard dialog. For instance, user inputs for dialog **740***b* and **740***c* are summarized by rule summary #1 (shown at **777**). The user can backtrack through the wizard panes, if needed, for modifying the new rule. Once the user is satisfied with the definition for the new rule, the user selects Finish button **780** for adding the new rule to the system. As shown in FIG. 7K, the interface **700** (now **700***a*) adds the new rule to the rule panel, at **781**. General information for the rule is provided in the rule details panel, as shown at **783**.

D. Internal methodologies

1. General

Detailed internal operation of the system of the present invention is perhaps best described by dividing the internal operation into the following general methods.

I. Loading the Client Monitor

II. Client Monitor did not find Supervisor (operation outside of LAN)

III. Unloading the Client Monitor (normally at workstation shutdown)

IV. Loading the Client Monitor in an ISP environment

V. Interpretation of a typical HTTP "GET" request

VI. Bandwidth and interactive use monitoring

VII. Managing network congestion

VIII. Intercepting WinSock messages

IX. Transmitting messages to Ring **3**

Each will now be described in turn.

2. Loading the Client Monitor

As illustrated in FIG. **8**, a method **800** for loading the Client Monitor component comprises the following steps. At step **801**, the Client Monitor checks if a Supervisor has been assigned for this Client Monitor. If a Supervisor has been assigned (i.e., yes), the Client Monitor sends a login request to the Supervisor, at step **802**. At step **803**, the Supervisor checks if the request comes from within the LAN. Then, at step **804**, the Supervisor checks if the Client Monitor (computer/user) has any Internet access rights; also, the Supervisor determines the department or workgroup for the Client Monitor, as indicated at step **805**.

Based on these determinations, the Supervisor filters rules appropriate for the client (i.e., application, Host, and other rules), and transmits them to the Client Monitor, at step **806**. Thereafter, at step **807**, the Client Monitor confirms successful reception of rules. The Client Monitor saves a copy of the rules onto a local hard disk (i.e., to a local storage medium), if available, as shown at step **808**. The Supervisor contacts the Firewall, at step **809**, to request Internet access for the Client Monitor. Connection between the Client Monitor and the Supervisor remains open; this is indicated at step **810**. Now, the Supervisor regularly sends check messages to the Client Monitor, as shown at step **811**. The Client Monitor can now store log information on local storage (if available), at step **812**. In a complementary fashion, the Client Monitor sends log messages to the Supervisor, at step **813**. If the Supervisor detects any problem with the Client Monitor, it notifies the Firewall to disable Internet access for the Client Monitor, as indicated by step **814**.

3. Client Monitor unable to locate Supervisor

As illustrated in FIG. **9**, a method **900** for handling the scenario of when the Client Monitor is unable to locate a Supervisor (operation outside of the LAN) comprises the following steps. At step **901**, the Client Monitor loads the last stored application, Host, rules database, and the like from local storage. At step **902**, the Client Monitor attempts to contact the Internet directly (for example via a dialup connection), but the last-stored rules continue to apply.

4. Unloading the Client Monitor

As illustrated in FIG. **10**, a method **1000** for unloading the Client Monitor (normally at workstation shutdown) comprises the following steps. At step **1001**, the Client Monitor component or application notifies the Supervisor that it is about to be unloaded. At step **1002**, the Supervisor contacts the Firewall of that Client Monitor to stop Internet access for that Client Monitor. At step **1003**, the Client Monitor stores any remaining log information on local storage (if available). At step **1004**, the Client Monitor sends any remaining log messages to the Supervisor. At step **1005**, the Client Monitor shuts down.

5. Unloading the Client Monitor

As illustrated in FIGS. **11**A-B, a method **1100** for loading the Client Monitor in an Internet Service Provider (ISP) environment comprises the following steps. At step **1101**, the RAS calls the ISP POP server using SLIP, PPP or similar protocol with user ID/password. At step **1102**, the ISP POP Server calls the ISP Authentication Server with user ID/password. At step **1103**, the ISP Authentication Server checks user ID and password. If these are valid, the Authentication Server checks with the ISP Supervisor if the user has access rules (mechanism) installed, at step **1104**. If rules are install, the ISP Authentication Server notifies ISP POP that client has Internet access restricted to the ISP "Sandbox" Server. Otherwise, the ISP Authentication Server notifies the

ISP POP that the client has unrestricted Internet access. At step **1105**, the Client Monitors send login requests to the ISP Supervisor. The Supervisor then transmits access rules and the like to the Client Monitor at step **1106**. At step **1107**, the Client Monitor saves a copy of the rules on a local hard disk to a local storage medium, if available. At step **1108**, the ISP Supervisor contacts the ISP POP Server to remove "sandbox" restrictions. At step **1109**, the connection between the Client Monitor and the Supervisor remains open. At step **1110**, the ISP Supervisor regularly sends check messages to the Client Monitor. At step **1111**, the Client Monitor stores log information on local storage (if available). At step **1112**, the Client Monitor sends log messages to the ISP Supervisor. If the ISP Supervisor determines any problems with the Client Monitor, it notifies the ISP POP Server to restrict access rights to "sandbox mode" at step **1113**.

6. Interpreting protocol commands (e.g., HTTP requests)

FIGS. **12**A–C illustrates a method **1200** for interpreting protocol commands, such as a typical HTTP "GET" request. At any time during the method, the Client Monitor can fail or redirect a call; it also can show the user a warning dialog but permit the command or request call to continue unchanged. The method **1200** comprises the following steps. At step **1201**, the application calls WinSock WSAStartup0 API command. At step **1202**, the Client Monitor intercepts the call and checks the rules and application database to see if the application or a specific version of the application has Internet access rights. If not, the Client Monitor fails the WSAStartup0 call at step **1203**. At step **1204**, the Application invokes socket0. At step **1205**, the Client Monitor intercepts the call and checks if the application or user has the right to continued use of the Internet (see also Bandwidth and interactive use monitoring). If not, the Client Monitor fails the socket0 call at step **1206**. At step **1207**, the Application contacts the Host using WinSock connect0. At step **1208**, the Client Monitor intercepts the call and checks the rules and Host database to see if the application has access rights to the specific Host. At step **1209**, the Client Monitor checks if the application or user has the right to continued use of the Internet (see also Bandwidth and interactive use monitoring). If the answer is no at steps **1208** and **1209**, the Client Monitor fails or redirects the connect0.call at step **1210**. At step **1211**, the Application calls WinSock send0 with the HTTP get command (e.g., "GET foo.html"). At step **1212**, the Client Monitor intercepts the call and determines the protocol based on a combination of the TCP/IP port address, the address family, contents, and the like. At step **1213**, the Client Monitor checks the rules and application database to see if the Application has the right to use HTTP. At step **1214**, the Client Monitor checks the rules database to see if the user/computer has the right to download ".html" files. If the answer is no at steps **1213** or **1214**, the Client Monitor fails or redirects the send0 call at step **1215**.

At step **1216**, the Client Monitor loads the content driver for ".html" files. At step **1217**, the application invokes WinSock recv0. At step **1218**, the Host sends the contents of "foo.html". At step **1219**, the Client Monitor intercepts the return of the recv0 call and passes the contents to the content driver. At step **1220**, the content driver parses the contents of "foo.html" and checks for the following components: (a) References to Java™, ActiveX, and the like (<APPLET> or <OBJECT> tags); (b) References to Netscape style plug-ins (<EMBED> tag); (c) Imbedded scripts such as Java Script™, VBScript, and the like (<SCRIPT> tag); (d) References to other files or components (<A HREF=\f "Symbol">, or <IMG SRC=\f "Symbol"> tags); and (e) Other syntax elements that are known or suspected to cause security or network problems. At step **1221**, the Contents driver checks the application and rules database to see if the specific HTML component is permissible. If it is not permissible, the driver either removes the HTML component or fails the recv0 call depending on the violated rule at step **1222**. At step **1223**, the Application received (sometimes modified) contents of "foo.html". At step **1224**, the Client Monitor intercepts the file I/O calls from the application and tries to determine where (if at all) the Application has saved the file it just received.

7. Bandwidth and interactive use monitoring

FIGS. **13**A–B illustrate a method **1300** for bandwidth and interactive use monitoring. During the method, the Client Monitor maintains a list of active Applications (processes) with various fields to track activities and a global activity record. The method **1300** comprises the following steps. At step **1301**, the Application calls WinSock send0 or recv0 calls. At step **1302**, the Client Monitor intercepts these calls and: (a) Marks the time of the call in a LastInternetAccess field of the Application's list entry; (b) Checks if the send0 or recv0 uses an Internet protocol usually associated with interactive activity (HTTP—Web browsing, online games, and the like); (c) If such a protocol is used, marks the time of the call in a LastInteractiveAccess field of the Application's list entry; (d) Adds the data lengths to a DataIn or DataOut accumulator or counter of the Application's list entry and global activity record; (e) If the DataIn or DataOut fields exceed rule-based quantity either for the specific Application or the user/workstation, the Client Monitor disables future Internet access and/or warns the user.

At step **1303**, the operating system, Windows, sends certain keyboard (WM_KEY), and mouse (WM_?BUTTONDOWN) messages to a window. At step **1304**, the Client Monitor intercepts these messages. At step **1305**, the Client Monitor identifies the target window and Application of the message and marks the time of the LastInteractiveUse field of the Application's list entry. At step **1306**, every minute the Client Monitor checks each entry of the Application list as follows: (a) Has the LastInternetAccess field changed in the last minute; (b) If yes, add one minute to a TotalInternetUse field of the Application's list entry; (c) Have the LastInteractiveAccess and LastInteractiveUse fields changed in the last five minutes; (d) If yes, add one minute to a TotalInteractiveUse field of the Application's list entry; (e) If the TotalInternetUse or TotalInteractiveUse fields of any Application's list entry have changed, also add one minute to the corresponding field of the global record; and (f) If any of the TotalInternetUse or TotalInteractiveUse fields exceed rule-based quantity either for the specific Application or the user/workstation, the Client Monitor disables future Internet access and/or warns the user.

8. Managing network congestion

As illustrated in FIG. **14**, a method **1400** for managing network congestion comprises the following steps. At step **1401**, if the Supervisor determines that congestion exists for Internet access (either by interpreting the log messages from the Client Monitors, its own monitoring of access speed, or third party monitoring tools), it notifies the Client Monitors of temporary access restrictions. At step **1402**, depending on the specific rules in force, the individual Client Monitors can either: (a) Delay Internet access for non-critical Applications or Protocols by: (i) Applications call WinSock send0, recv0, connect0 calls, and the like, (ii) the Client Monitor intercepts the call and checks the rules and Application database if calls relate to non-critical activities, (iii) If yes—delay the specific thread of the Application by a predetermined amount (e.g., 10 seconds), or (iv) open bandwidth for critical activities; or (b) Disable Internet Access for non-critical Applications or protocols.

9. Intercepting communication messages (e.g., WinSock messages)

FIGS. **15**A–B illustrate a method **1500** for intercepting communication driver (e.g., WinSock) messages. The following method description focuses on a Windows 95 implementation with the following standard Microsoft WinSock component: Wsock32.dll and Wsock.vxd. The implementation is similar under Windows NT and other operating systems.

The method operates as follows. At step **1501**, the Client Monitor loads the Client VxD (Windows virtual driver file). At step **1502**, the Client VxD loads the WinSock virtual driver file, Wsock.vxd, and redirects the WinSock DeviceIOControl code pointer of Wsock.vxd to its own interception routine. At step **1503**, the application calls the WinSock function in the WinSock dynamic link library, Wsock32.dll, that requires Internet access. At step **1504**, Wsock32.dll processes the parameters and calls Wsock.vxd via the the Windows Win32 DeviceIoControl0 function call. At step **1505**, the Client VxD looks up the call via an "intercept before" dispatch table. At step **1506**, if the dispatch table requires an intercept, the Client VxD creates an interception message and calls the Client Monitor. At step **1507**, if the Client Monitor allows the call to go forward, the Client VxD calls the original Wsock.vxd routine, otherwise it returns Wsock32.dll and the Application. At step **1508**, the Client VxD looks up the call via the "intercept after" dispatch table. If the dispatch table requires an intercept, the Client VxD creates an interception message and calls the Client Monitor at step **1509**. At step **1510**, the Client VxD returns to Wsock32.dll with either the original return results or results modified by the Client Monitor.

10. Transmitting messages from Ring **0** to Ring **3**

FIGS. **16**A–B illustrate a method **1600** for transmitting messages from one memory protection ring to another (e.g., from highly-privileged Ring **0** to lesser-privileged Ring **3**). Ring **3** and Ring **0** refer to execution protection rings available on Intel-based computers (e.g., having Intel 80386 and later CPUs) and specify the application execution mode (Ring **3**) or the kernel execution mode (Ring **0**). Ring **3** programs and their components have a separate memory address space for each application. Ring **0** components (VxDs), on the other hand, share a common memory space. One can create a pointer that is accessible both from Ring **0** and all Ring **3** processes.

A difficult task is forwarding intercepted data from Ring **0** to Ring **3**, and later applying any modifications that the Ring **0** component has made to the call parameters and return results. Difficulties include: (1) the intercept occurs in the memory context of the Application's process, thus one needs to wait until the process received its time share from Windows; and (2) the call parameters (memory pointers) are valid only in the context of the Application's process. The Client VxD and the Client Monitor share a common array of messages and two pointers into that array. The Client VxD adds a message to that array and the Client Monitor picks it up using a standard first-in/first-out (FIFO) approach. Each fixed size message record has some space for additional data, if that size is not sufficient the additional data (e.g., recv0 or send0 data) is mapped into global memory space so it is accessible for the Client Monitor's process. Because the events occur asynchronously, particular care is needed to avoid situations where Windows switches the executing thread at critical sections, leading to possible deadlocks.

Messages are generated from the following sources (intercepted WinSock calls).

1. Threads created and destroyed (to keep track of Applications); and

2. File activities (using the ring **0** IFSMgr_InstallFileSystemApiHook mechanism)

Some messages require that the Client Monitor processes the message (and possibly stop or modify the underlying activities) before the Application's process can be allowed to continue, while other messages are just for information purposes.

The method **1600** comprises the following steps. At step **1601**, File, WinSock, or Thread components of Client VxD call a message dispatcher, "Dispatcher." At step **1602**, the Dispatcher determines if any additional data is required. If additional data is required, the Dispatcher determines if additional data fits into extra space, and then copies data into the additional space. If additional data is not required, the Dispatcher determines if data is already mapped into global memory space—if it is not, the Dispatcher allocates a global memory pointer to data and puts the pointer into the message body. At step **1603**, the Dispatcher copies the message to an array. At step **1604**, the Dispatcher determines if it needs to wait for message processing because: (a) It might need to free the global memory pointer; (b) the Client Monitor needs to approve the underlying action; or (c) the Client Monitor might patch any of the parameters.

At step **1605**, if the Dispatcher needs to wait, it: (a) tells the operating system (e.g., Windows) to switch to the Ring **3** Client Monitor's message thread; or (b) puts itself (and therefore the application thread) into sleep mode, otherwise it returns immediately to the caller. After the Client Monitor has processed the message, the Dispatcher does one or more of the following actions at Step **1606**: (a) De-allocates the global memory pointer, if previously allocated; and/or (b) Copies any patched memory to correct data.

While the invention is described in some detail with specific reference to a single-preferred embodiment and certain alternatives, there is no intent to limit the invention to that particular embodiment or those specific alternatives. Thus, the true scope of the present invention is not limited to any one of the foregoing exemplary embodiments but is instead defined by the appended claims.

What is claimed is:

1. In a system comprising a plurality of client computers connected to a network and having Internet access, a method for managing Internet access for a particular client computer, the method comprising:

providing at the particular client computer a client monitoring process;

providing at another computer on the network a supervisor process, said supervisor process specifying rules which govern Internet access by the client computers;

transmitting at least a subset of said rules to the particular client computer;

at the client monitoring process, trapping a request for Internet access from the particular client computer; and

processing the request for Internet access by performing substeps of:

(i) determining whether the request for Internet access violates any of the rules transmitted to the particular client computer, and

(ii) if the request for Internet access violates any of the rules transmitted to the particular client computer, denying the request for Internet access.

2. The method of claim **1**, wherein the particular client computer includes a communication driver for processing requests for Internet access and wherein said step of providing at the particular client computer a client monitoring process comprises:

providing at the particular client computer a process which traps at the communication driver requests for Internet access.

3. The method of claim **1**, wherein said step of providing at the particular client computer a client monitoring process includes:

installing at the particular client computer a client monitoring process which executes anytime the communication driver processes requests for Internet access.

4. The method of claim **1**, wherein said another computer includes a server computer connected to the network.

5. The method of claim **1**, wherein said another computer includes another client computer connected to the network.

6. The method of claim **1**, wherein said rules transmitted to the particular client computer specify whether the particular client computer is allowed any Internet access.

7. The method of claim **1**, wherein said rules transmitted to the particular client computer specify which applications are allowed Internet access.

8. The method of claim **1**, wherein said rules transmitted to the particular client computer specify a particular type of Internet access which is allowed.

9. The method of claim **1**, wherein said system includes a "firewall" application for selectively blocking Internet access and wherein said substep of denying the request for Internet access includes:

instructing said firewall application to block Internet access for the particular client computer.

10. The method of claim **9**, wherein said firewall application operates independently to block Internet access for any client according to rules specified for the firewall application.

11. The method of claim **1**, wherein said rules which govern Internet access by the client computers include rules which are enforced against selected ones of users, computers, and groups thereof.

12. The method of claim **11**, wherein said transmitting at least a subset of said rules step includes:

determining, based on identification of users, computers, or groups thereof and for which rules have been defined, a subset of said rules filtered for a given user at the particular client computer.

13. The method of claim **12**, wherein said users are identified by user name.

14. The method of claim **12**, wherein said computers are identified by Internet Protocol (IP) addresses.

15. The method of claim **1**, wherein said transmitting at least a subset of said rules to the particular client computer includes:

transmitting a set of default rules for the particular client, if no particular rules are already defined for the client.

16. In a system comprising a plurality of client computers connected to a network and having Internet access, a method for managing Internet access for a particular client computer on a per application basis, the method comprising:

storing at a supervisor computer a list of applications and versions thereof defining which applications are permitted Internet access;

transmitting said list from the supervisor computer to the client computer;

at the client computer, trapping a request for Internet access from a particular application;

based on said list, determining whether the request for Internet access is from an application or version thereof which is permitted Internet access; and

if the request for Internet access is from an application or version thereof which is not permitted Internet access, blocking Internet access for the application.

17. The method of claim **16**, wherein said storing step includes:

storing with a supervisor process executing on a server computer connected to the network the list of applications and versions thereof which are permitted Internet access.

18. The method of claim **16**, wherein said storing step includes:

storing with a supervisor process executing on another client computer connected to the network the list of applications and versions thereof which are permitted Internet access.

19. The method of claim **16**, wherein said list includes executable names and version numbers for applications which are permitted Internet access.

20. The method of claim **16**, wherein said list includes executable names and version numbers for applications which are not permitted Internet access.

21. The method of claim **16**, wherein said list includes Internet access activities which are permitted or restricted for applications or versions thereof.

22. The method of claim **21**, wherein Internet access activities comprise use of particular communication protocols.

23. The method of claim **22**, wherein said communication protocols include at least one of Hypertext Transport Protocol (HTTP) and File Transport Protocol (FTP).

24. The method of claim **22**, wherein said communication protocols include an e-mail protocol.

25. The method of claim **16**, wherein said Internet access activities comprise at least one of browsing activity and e-mail activity.

26. A computer system regulating access by client computers comprising:

a plurality of client computers which can connect to at least one open network;

supervisor means provided at a computer which is in communication with each client computer to be regulated, said supervisor means including a database of enforcement rules governing access of client computers to said at least one open network;

means for transferring rules from the database of enforcement rules to each computer requiring access to said at least one open network and which is to be regulated; and

monitoring means provided at each client computer which is to be regulated, for selectively blocking access to said at least one open network based on said transferred rules.

27. The system of claim **26**, further comprising:

means for selectively blocking access based on properties of applications executing at said client computers which attempt access to said at least one open network.

28. The system of claim **27**, wherein said properties of applications include version and executable name for each application.

29. The system of claim **27**, wherein said properties of applications include types of activities which applications are either allowed to perform or restricted from performing.

30. The system of claim **29**, wherein said types of activities include at least one of using e-mail and browsing.

* * * * *

US006950947B1

(12) United States Patent
Purtell et al.

(10) Patent No.: **US 6,950,947 B1**

(45) Date of Patent: **Sep. 27, 2005**

(54) SYSTEM FOR SHARING NETWORK STATE TO ENHANCE NETWORK THROUGHPUT

(75) Inventors: **Andrew Purtell**, Northridge, CA (US); **Roger Knobbe**, Torrance, CA (US); **Stephen Schwab**, Manhattan Beach, CA (US)

(73) Assignee: **Networks Associates Technology, Inc.**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1020 days.

(21) Appl. No.: **09/597,973**

(22) Filed: **Jun. 20, 2000**

(51) **Int. Cl.**^{7} **G06F 11/30**

(52) **U.S. Cl.** **713/201**; 713/152; 370/229; 370/231; 370/235; 370/352

(58) **Field of Search** 713/200–201, 713/152; 709/224–229, 236; 370/229–235, 370/352–356, 252, 253, 254, 255, 256

(56) References Cited

U.S. PATENT DOCUMENTS

5,737,531	A	*	4/1998	Ehley	709/208
5,828,846	A	*	10/1998	Kirby et al.	709/238
5,941,988	A		8/1999	Bhagwat et al.	713/201
6,006,268	A	*	12/1999	Coile et al.	709/227
6,119,167	A	*	9/2000	Boyle et al.	709/234
6,295,557	B1	*	9/2001	Foss et al.	709/224
6,298,380	B1	*	10/2001	Coile et al.	709/227
6,442,565	B1	*	8/2002	Tyra et al.	707/102
6,473,406	B1	*	10/2002	Coile et al.	370/248

FOREIGN PATENT DOCUMENTS

EP	0909072		4/1999	.... H04L 29/06
EP	0909073		4/1999	.... H04L 29/06
EP	909073	A2 *	4/1999	.... H04L 29/06
EP	0909074		4/1999	.... H04L 29/06
EP	0910197		4/1999	.... H04L 29/06
EP	0944209		9/1999	.... H04L 12/56
WO	WO9854644		12/1998	.... G06F 11/00
WO	WO0000879		1/2000	
WO	WO0002114		1/2000	.... G06F 1/00

OTHER PUBLICATIONS

Schilke, Andreas. "TCP over Satelite Links". http//www.t-kn.tu-berlin.de/curricula/ss97/bnt97/schilke.html. Jun. 5, 1997.*

"Advanced Security Proxies: An Architecture and Implementation for High-Performance Network Firewalls," Roger Knobbe, Andrew Purtell, and Stephen Schwab, *Proceedings of the DISCEX 2000 Conference,* IEEE Computer Society, Jan. 2000.

RFC 2140, "TCP Control Block Interdependence," Joe Touch, Apr. 1997.

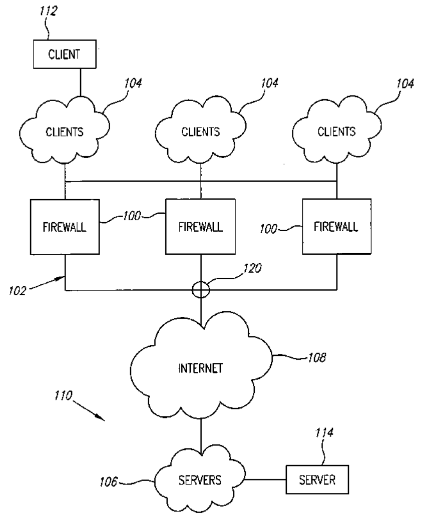
* cited by examiner

Primary Examiner—Andrew Caldwell
Assistant Examiner—Paul Callahan
(74) *Attorney, Agent, or Firm*—Zilka-Kotab, PC; Christopher J. Hamaty

(57) ABSTRACT

Two or more computers acting as firewalls share network state data to enhance throughput performance. A firewall creates a separate common TCP control block (CCB) for each group of TCP connections through the firewall having common endpoints. The CCB is a shared data structure comprising a single microstate shared across the group of TCP connections. Each such individual TCP connection has a TCP control block, which instead of a microstate, contains a pointer to the appropriate CCB. Preferably, each firewall receives CCBs from its peers and stores them. Each firewall preferably adjusts data traffic passing through it based on the CCBs stored within it. By adjusting traffic to reduce or eliminate congestion, throughput is enhanced.

12 Claims, 6 Drawing Sheets



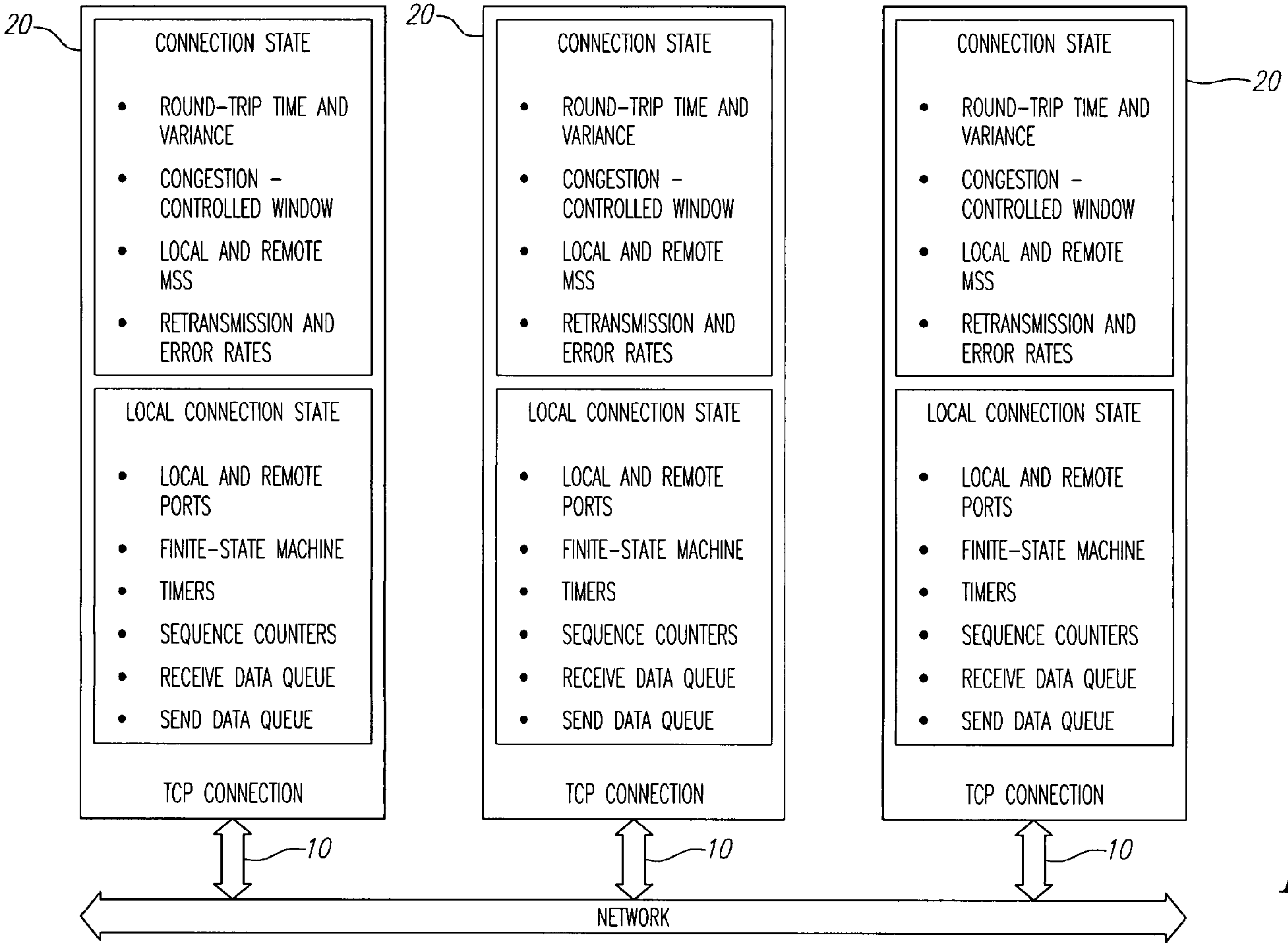


FIG. 1
(PRIOR ART)

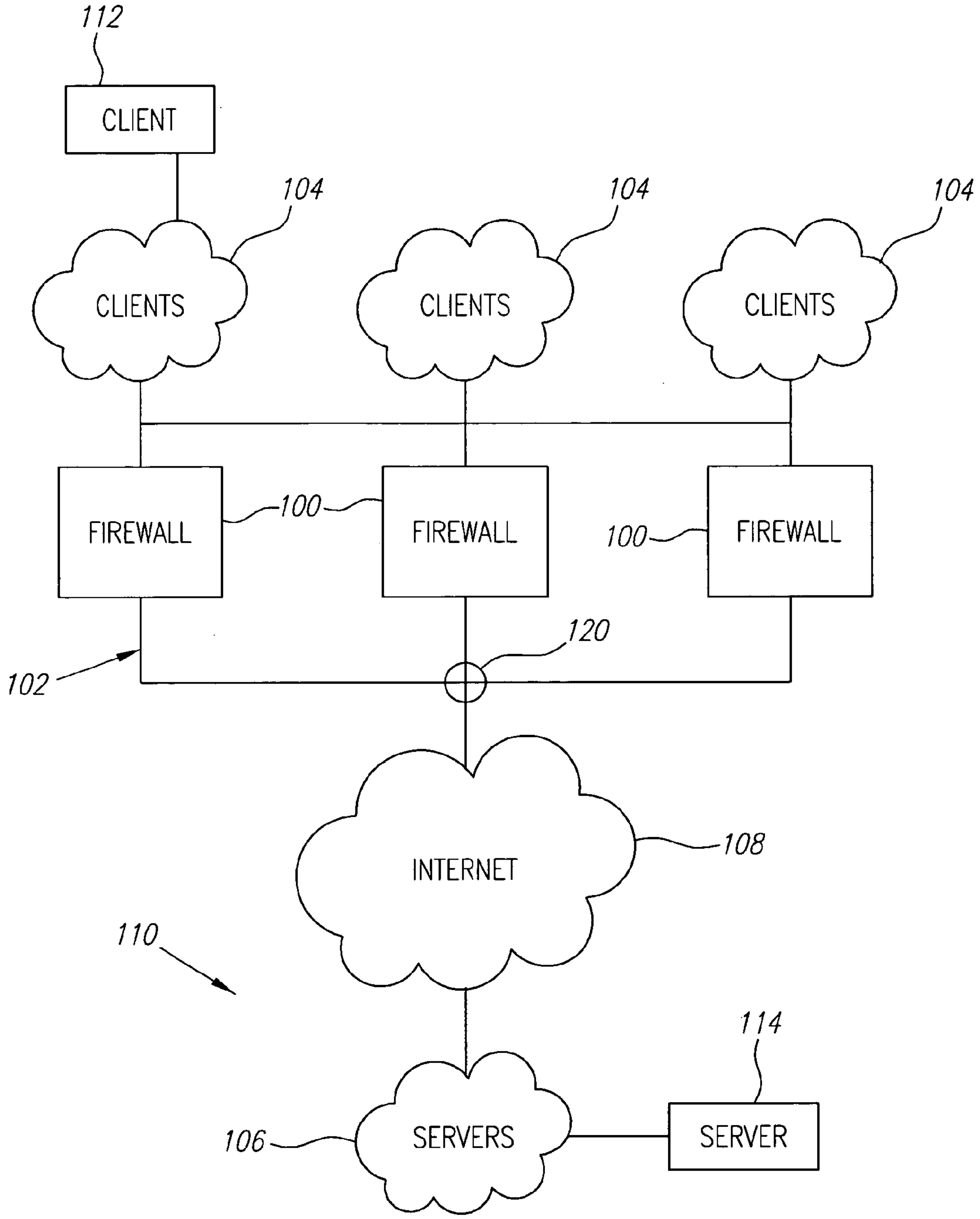

112
CLIENT
104
CLIENTS
104
CLIENTS
104
CLIENTS
FIREWALL
100
FIREWALL
100
FIREWALL
102
120
INTERNET
108
110
114
106
SERVERS
SERVER

FIG. 2

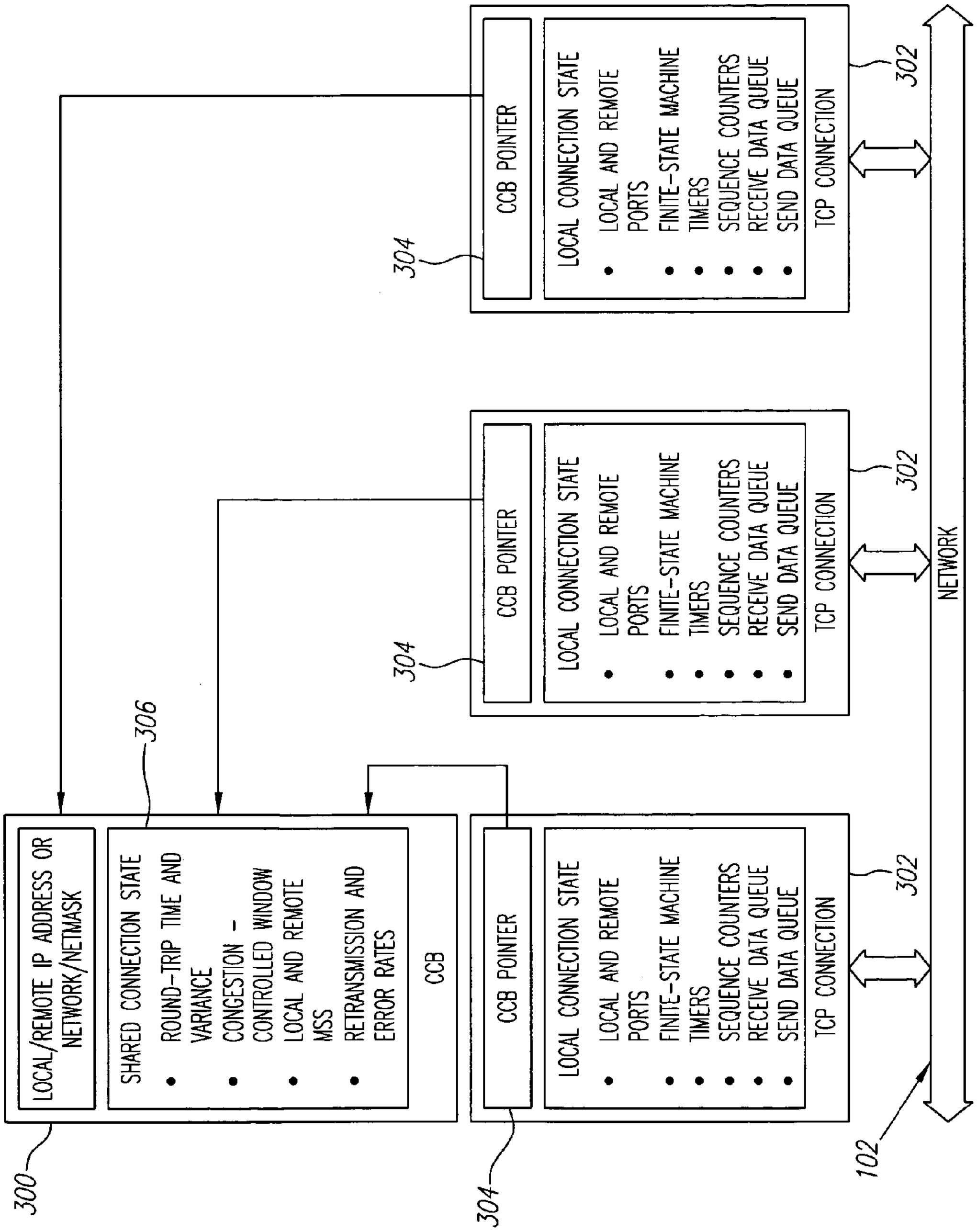

LOCAL/REMOTE IP ADDRESS OR NETWORK/NETMASK
SHARED CONNECTION STATE
• ROUND-TRIP TIME AND VARIANCE
• CONGESTION - CONTROLLED WINDOW
• LOCAL AND REMOTE MSS
• RETRANSMISSION AND ERROR RATES
CCB
306
300
CCB POINTER
304
LOCAL CONNECTION STATE
• LOCAL AND REMOTE PORTS
• FINITE-STATE MACHINE
• TIMERS
• SEQUENCE COUNTERS
• RECEIVE DATA QUEUE
• SEND DATA QUEUE
TCP CONNECTION
302
CCB POINTER
304
LOCAL CONNECTION STATE
• LOCAL AND REMOTE PORTS
• FINITE-STATE MACHINE
• TIMERS
• SEQUENCE COUNTERS
• RECEIVE DATA QUEUE
• SEND DATA QUEUE
TCP CONNECTION
302
CCB POINTER
304
LOCAL CONNECTION STATE
• LOCAL AND REMOTE PORTS
• FINITE-STATE MACHINE
• TIMERS
• SEQUENCE COUNTERS
• RECEIVE DATA QUEUE
• SEND DATA QUEUE
TCP CONNECTION
302
NETWORK
102

FIG. 3

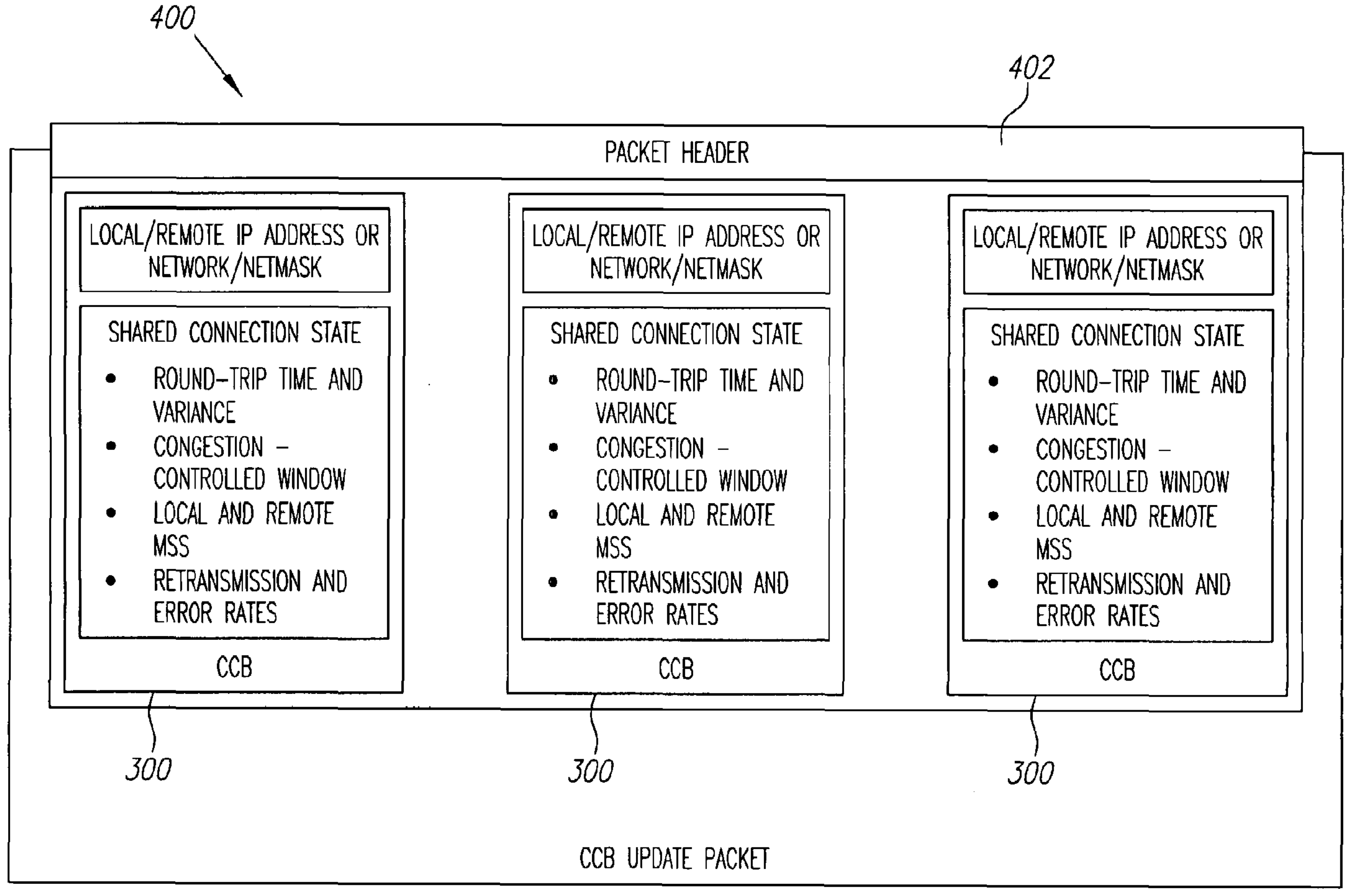

400
402
PACKET HEADER
LOCAL/REMOTE IP ADDRESS OR NETWORK/NETMASK
SHARED CONNECTION STATE
• ROUND-TRIP TIME AND VARIANCE
• CONGESTION - CONTROLLED WINDOW
• LOCAL AND REMOTE MSS
• RETRANSMISSION AND ERROR RATES
CCB
300
LOCAL/REMOTE IP ADDRESS OR NETWORK/NETMASK
SHARED CONNECTION STATE
• ROUND-TRIP TIME AND VARIANCE
• CONGESTION - CONTROLLED WINDOW
• LOCAL AND REMOTE MSS
• RETRANSMISSION AND ERROR RATES
CCB
300
LOCAL/REMOTE IP ADDRESS OR NETWORK/NETMASK
SHARED CONNECTION STATE
• ROUND-TRIP TIME AND VARIANCE
• CONGESTION - CONTROLLED WINDOW
• LOCAL AND REMOTE MSS
• RETRANSMISSION AND ERROR RATES
CCB
300
CCB UPDATE PACKET

FIG. 4

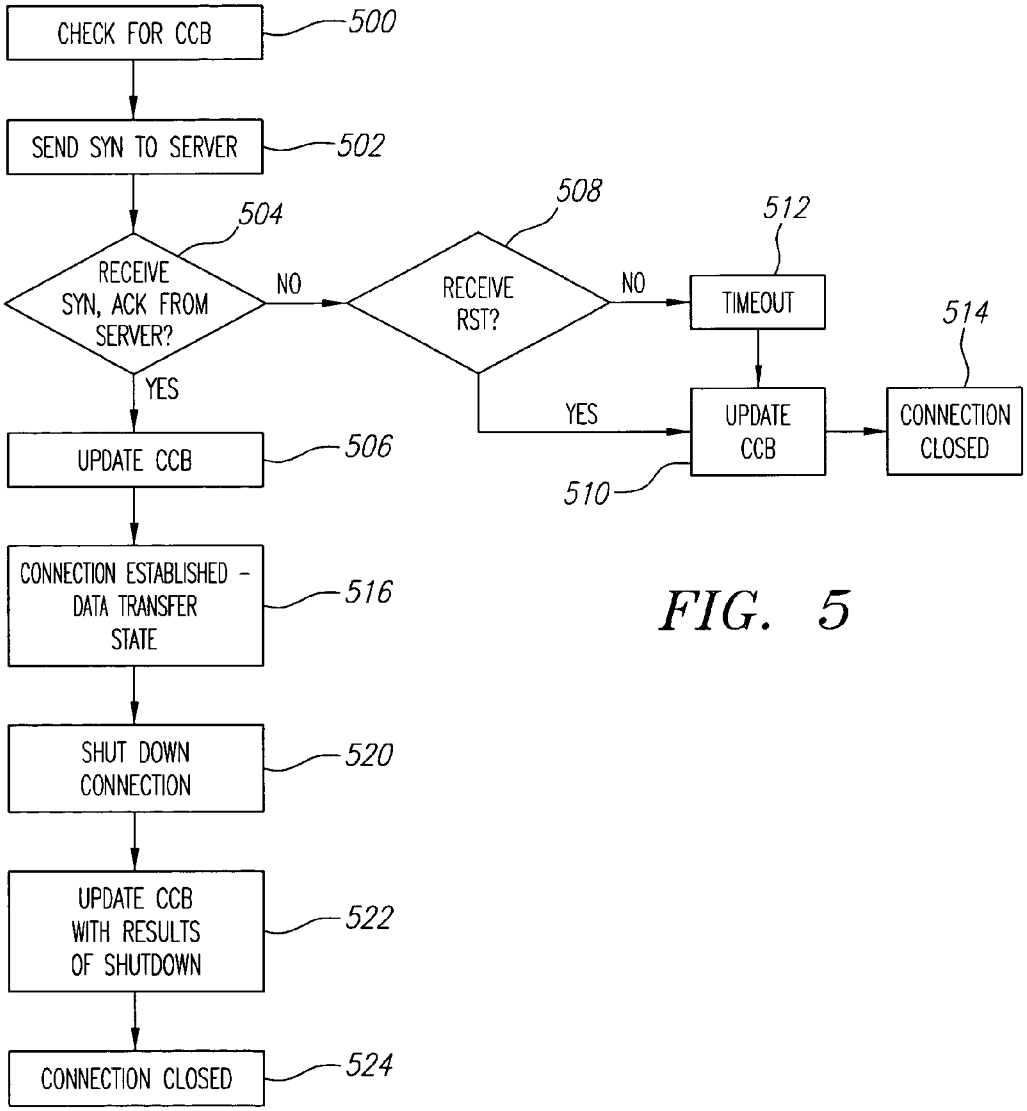

CHECK FOR CCB
500
SEND SYN TO SERVER
502
RECEIVE SYN, ACK FROM SERVER?
504
NO
YES
RECEIVE RST?
508
NO
YES
TIMEOUT
512
UPDATE CCB
510
CONNECTION CLOSED
514
UPDATE CCB
506
CONNECTION ESTABLISHED - DATA TRANSFER STATE
516
SHUT DOWN CONNECTION
520
UPDATE CCB WITH RESULTS OF SHUTDOWN
522
CONNECTION CLOSED
524

FIG. 5

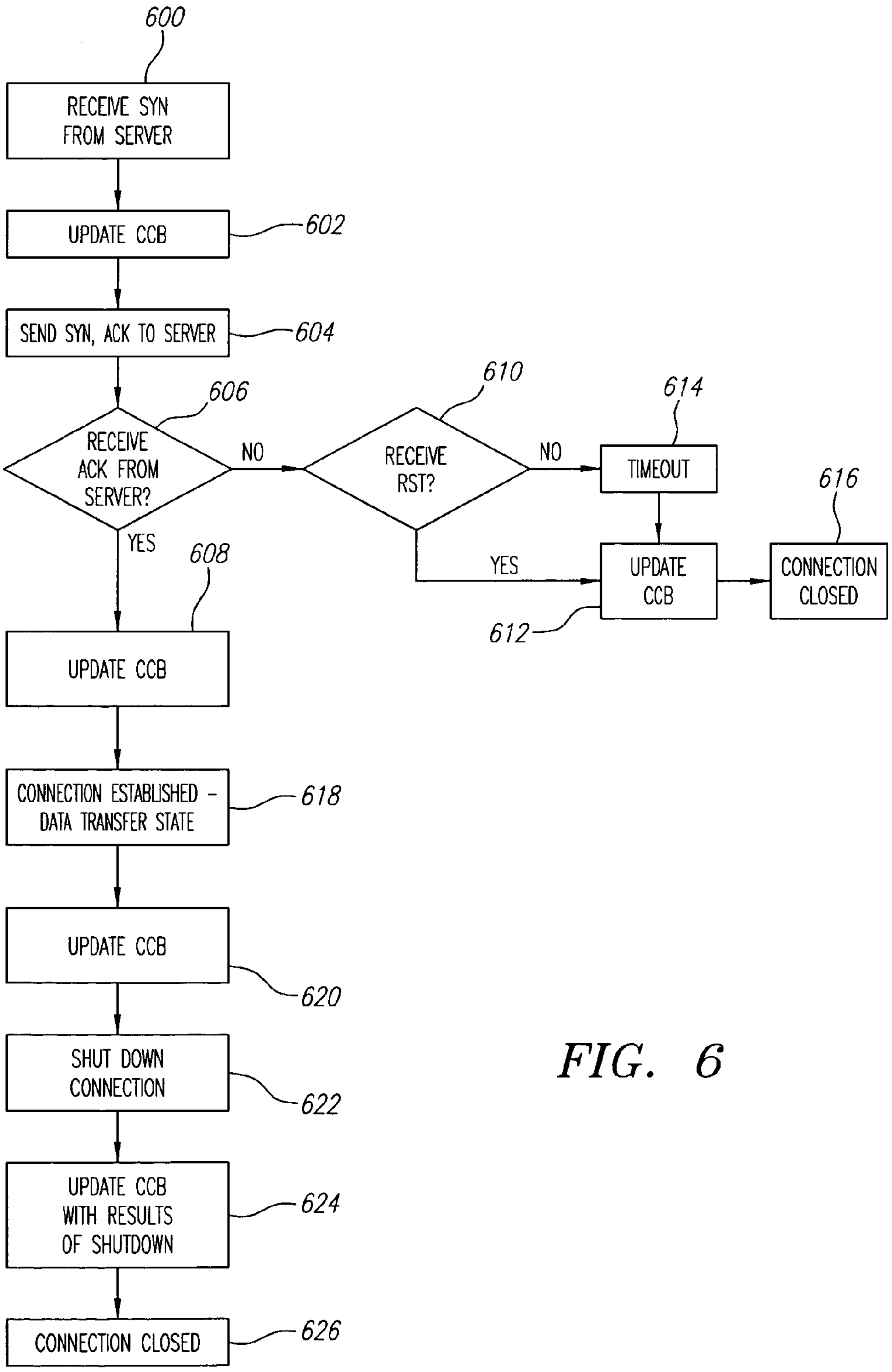

600
RECEIVE SYN FROM SERVER
UPDATE CCB
602
SEND SYN, ACK TO SERVER
604
606
RECEIVE ACK FROM SERVER?
NO
610
RECEIVE RST?
NO
614
TIMEOUT
616
YES
UPDATE CCB
CONNECTION CLOSED
612
YES
608
UPDATE CCB
CONNECTION ESTABLISHED - DATA TRANSFER STATE
618
UPDATE CCB
620
SHUT DOWN CONNECTION
622
UPDATE CCB WITH RESULTS OF SHUTDOWN
624
CONNECTION CLOSED
626

FIG. 6

SYSTEM FOR SHARING NETWORK STATE TO ENHANCE NETWORK THROUGHPUT

The U.S. Government has a paid-up license in this invention and the right in limited circumstances to require the patent owner to license others on reasonable terms as provided by the terms of contract no. F30602-97-C-0336 awarded by the Defense Advanced Research Projects Agency.

BACKGROUND OF THE INVENTION

The field of invention is computer networking, and more specifically a system for sharing network state among related TCP connections to enhance network throughput.

Transmission Control Protocol (TCP) is a common and well-known protocol for transmitting data between computers over a network such as the Internet. The TCP standard is defined in RFC 793, Transmission Control Protocol. TCP divides a file or other data into packets, numbers the packets, then transmits them over a network. TCP keeps track of these packets, and reassembles them at their destination back into their original configuration. TCP is a connection-oriented protocol, which means that a connection is established between the machine transmitting data and the machine receiving data, and that connection is maintained until all of the data to be exchanged between the machines has been transmitted. The TCP standard is almost universally used in transmitting data across the Internet between a server and a client.

Data transmission over a network creates security concerns. A common method of combating threats to the security of a computer, or the security of an internal network such as a corporate network, is to provide one or more machines known as firewalls between the internal network and an external network. A firewall typically filters network packets received from the external network to determine whether to forward them to their destination on the internal network. A common type of firewall is a proxy server that makes requests to the external network on behalf of users of the internal network. For example, a user on the internal network may wish to browse a web page. Rather than connecting directly to the server, the user would connect to the proxy server, which would request the web page from the external server on behalf of the user, then examine the data served based on the access control policy implemented by that firewall. This process is typically invisible to the user.

A firewall enforces an access control policy between two networks. The firewall makes a policy decision for an individual connection based on information limited to that single connection. This information may include packet contents, source and/or destination addresses and ports, or other information. However, information regarding overall network conditions is not available at the firewall during connection setup, which is where the policy decision must be made. Instead, each TCP connection **10** has a separate TCP control block **20** that is not communicated between individual connections, as shown in FIG. **1**. The TCP control block **20** is known in the art, and includes state information for a single TCP connection **10**. Thus, current firewalls do not take advantage of traffic locality (i.e., repeated network transactions) between client/server pairs. Especially for proxy firewalls utilizing TCP, this may result in lower throughput than would otherwise be possible, due to the adverse impact of the slow-start data transmission mechanisms built into TCP. This process is time-consuming, relatively speaking, and typically results in decreased firewall throughput.

SUMMARY OF THE PREFERRED EMBODIMENTS

In the present invention, two or more firewalls share network state data to enhance throughput performance.

In an aspect of a preferred embodiment, a firewall creates a common TCP control block (CCB) for each group of TCP connections through the firewall having common endpoints. The CCB is a data structure that contains state data shared among the individual TCP connections within a specific group. Each individual TCP control block in a group contains a pointer to the CCB in lieu of the connection data stored within the CCB.

In another aspect of a preferred embodiment, a firewall receives CCBs from its peers and stores them for later use.

In another aspect of a preferred embodiment, each firewall adjusts data traffic passing through it based on the CCBs stored within it. By adjusting traffic to reduce or eliminate congestion, throughput is enhanced.

BRIEF DESCRIPTION OF DRAWINGS

FIG. **1** is a schematic representation of a prior art implementation of TCP control blocks.

FIG. **2** is a schematic representation of an internal network connected to and protected from an external network by a number of firewalls.

FIG. **3** is a schematic representation of TCP connection state sharing in a firewall adapted to utilize common TCP control blocks.

FIG. **4** is a schematic representation of a CCB update message.

FIG. **5** is a flow chart illustrating a process for utilizing a CCB when opening a connection from a firewall to an external network.

FIG. **6** is a flow chart illustrating a process for utilizing a CCB when receiving a request for connection at the firewall from an external network.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring to FIG. **2**, a diagram of networks interconnected by firewalls is shown. A number of firewalls **100** are connected together by an internal network **102**. A number of clients **104** access one or more servers **106** through an electronic network such as the Internet **108**, connected to the servers **106** through the firewalls **100**. The combination of the Internet **108** and the servers **106** accessible through it may be referred to as the external network **110**.

The term "client" is commonly used to describe a program or user requesting data in a client/server relationship, and may also be used to describe the physical hardware in which a client program is located. For the purposes of this specification, the combination of the hardware in which a client program is located and the client program itself will be referred to as a client **104**. The client **104** may include any device capable of connecting to a server **106** through an electronic network such as the Internet **108**, such as but not limited to a personal computer, a network appliance, or a personal digital assistant such as Palm Computing's Palm™ device. It is also within the scope of the preferred embodiment to treat a subnetwork as a client **104** in order to improve throughput to and from the subnetwork.

The term "server" is commonly used to describe a computer program that provides services to other computer programs in the same or other computers, and may also be used to describe the physical hardware in which a server program is located. In a client/server relationship, a server is used to provide data to a client that has requested that data. For the purposes of this specification, the combination of the hardware in which a server program is located and the server program itself will be referred to as a server **106**. Two connected computers may switch back and forth from acting as a client to acting as a server in order to effectuate data transfer.

Groups of clients **104** are shown in FIG. **2**, schematically representing that traffic between the clients **104** and the external network **110** is divided between the firewalls **100**. However, a particular client **112** is preferably not limited to the use of a single firewall **100**, and instead may connect to a different firewall **100** to connect to the external network **110**. The particular client **112** is a subset of the group of clients **104**, and is separated out for clarity. The clients **104** preferably are connected to one another behind the firewalls **100**. That is, communications between clients **104** need not pass through a firewall **100**.

It is within the scope of the preferred embodiment to provide a single firewall **100** between the clients **104** and the Internet **108**. This configuration is advantageously used in settings where there are a small number of clients **104**, or where access to the Internet **108** by the clients **104** is infrequent or the amount of data to be transferred to and from the internal network **102** is small.

In a preferred embodiment, the firewalls **100** are configured as proxy servers. The configuration and use of proxy servers as firewalls **100** is well known in the art. Typically, a proxy server receives a request from an individual client **112** for a specific resource located on a specific server **114** in the external network **110**. The particular server **114** is a subset of the group of servers **106**, and is separated out for clarity. The resource request is generally for a Web page, but other data may be requested by the individual client **112** as well. The proxy server then acts as a client on behalf of the individual client **112**, and uses one of its own IP addresses to request the resource from the specific server **114**. The specific server **114** then transmits that resource to the proxy server, which forwards the result to the individual client **112**. The existence and operation of the proxy server are generally invisible to the individual client **112**; all returned resources appear to the client **112** to have been transmitted directly from the specific server **114**.

Typically, there are a number of open TCP connections between the clients **104** and the servers **106** on the external network. Each TCP connection has an associated TCP control block that maintains the state of that TCP connection. The use of a TCP control block is known to those skilled in the art; the contents of such known TCP control blocks are shown in FIG. **1**. The TCP control block typically includes a local process state and a per-connection state. The local process state typically includes pointers to send and receive buffers, pointers to retransmission queue and current segment, and pointers to Internet Protocol (IP). The IP standard is well known to those skilled in the art. The per-connection state may be characterized as having a macrostate and a microstate. The macrostate describes the finite state machine, and includes the data needed to maintain that state, such as but not limited to endpoint numbers, timers, and flags. The microstate describes the protocol after a connection has been established, to maintain the reliability and congestion control of the data transferred in the connection. The microstate includes, but is not limited to, round trip time (RTT) and variance, congestion controlled window size and threshold, local and remote maximum segment size (MSS), retransmission and error rates, send and receive window state, and maximum windows seen. The characterization of the TCP control block contents as having a microstate, a macrostate and a local process state, and the contents and functions of these states, are known to those skilled in the art.

Referring to FIG. **3**, in a preferred embodiment, one common TCP control block (CCB) **300** is created by a firewall **100** for each group of connections between that firewall **100** and a particular server **114**. In a preferred embodiment, the CCB **300** comprises shared connection state data **306**, which preferably includes round trip time (RTT) and variance, the congestion-controlled window, the local and remote MSS, and retransmission and error rates. Preferably, the shared connection state data **306** includes the microstate data for the TCP connection between a given firewall **100** and a particular server **114**. The TCP control blocks **302** for each individual connection having a CCB **300** each include a pointer to the corresponding CCB **300** for shared connection state data **306**. That is, a single CCB **300** may be simultaneously utilized by a number of different TCP connections to retain shared connection state data **306** that is accessed via a pointer in the TCP control block **302** of each connection. As an example of the usage of CCBs **300**, if three connections were active between a firewall **100** and a search engine web site, and five connections were active between a firewall **100** and a business service web site, the firewall **100** would create and maintain two CCBs **300**: one CCB **300** for the connections between the firewall **100** and the search engine, and one CCB **300** for the connections between the firewall **100** and the business service web site. In the example above, the three TCP control blocks **302** for the connections between the firewall **100** and the search engine each contain a pointer to shared connection state data **306** in the corresponding CCB **300**, instead of individually containing the shared connection data **306** themselves.

The CCB **300** contains data, such as but not limited to round trip time and variance, that is substantially related to the physical connection between two entities connected via TCP. For example, round trip time will vary depending on the distance between and the network complexity and congestion between a firewall **102** and a particular server **114**. Thus, there is advantageously only a single point of contact **120** between the internal network **102** and the external network **110**. Further, the firewalls **100** are advantageously located in physical proximity to one another. Where there is a single connection **120** between the internal network **102** and the external network **114**, the state components that are related to the distance between and the congestion between the internal network **102** and a particular server **114** will be substantially identical for each firewall **100**. That is, the physical separation of the firewalls **100** will be negligible compared to the distance between the single point of contact **120** and the particular server **114**. Thus, the shared connection state data **306** that is that is a function of the distance between the firewall **100** and a particular server **114** will be substantially identical for each firewall **100**, and will be accurate and useful for each firewall **100**. However, it is within the scope of the preferred embodiment to use more than one point of contact between the internal network **102** and the external network **110**, or to physically separate the firewalls **100** over a distributed network, then use the shared CCBs **300** as a starting point or first approximation for a TCP connection. It is also within the scope of the preferred embodiment to utilize one or more CCBs **300** in a single firewall **100** having no network peers, or to use one or more CCBs **300** in a number of firewalls **100** without sharing CCBs **300** between firewalls **100**.

Each CCB **300** is preferably stored within the firewall **100** where it is used, in order to facilitate rapid access. However, the CCB **300** may be located in another location where it is accessible to the firewall **100**, such as a server or a dedicated computer connected to one or more of the firewalls **100** over the internal network **102** or through a dedicated connection.

In a preferred embodiment, each individual TCP connection sharing a single CCB **300** acts in turn as a lead connection for that CCB **300**. For example, where data transfer is simultaneously occurring over three TCP connections sharing a single CCB **300**, the first connection may be the initial lead connection. The CCB **300** may be updated due to state changes in the first connection. The next update to the CCB **300** would come from the second TCP connection, and after that update, the next update would come from the third TCP connection. The status of lead connection would then rotate back to the first connection. By updating the CCB **300** based on network conditions encountered by each of the TCP connections sharing it, the usefulness and accuracy of the CCB **300** for a particular client/server pair is increased.

Referring to FIG. **4**, a CCB update packet **400** is shown. In a preferred embodiment, the CCB update packet **400** is the data structure through which a firewall **100** shares its CCBs **300** with the other firewalls **100** on the internal network **102**. The CCB update packet **400** includes one or more CCBs **300**. The one or more CCBs **300** in the CCB update packet **400** are copies of the one or more CCBs **300** located in the computer which originated the CCB update packet **400**. The CCB update packet **400** includes a packet header **402** identifying the CCB update packet **400** and providing other information useful for properly routing the CCB update packet **400** to the appropriate firewall or firewalls **100**. The use of a packet header to transmit a data packet is known in the art.

A firewall **100** may share CCBs **300** with another firewall **100** on the internal network by pushing its own CCBs **300** to one or more network peers, or by pulling CCBs **300** from one or more network peers. If the firewall **100** pushes its CCBs **300** to a network peer, the firewall **100** makes a copy of the one or more CCBs **300** associated with its TCP connections, and transmits those CCBs **300** to one or more other firewalls **100** in a CCB update packet **400**. In a preferred embodiment, the firewall **100** pushes a CCB update packet **400** to its network peers on a periodic basis; that period is preferably fixed and preferably chosen to be within the range of one second to thirty seconds. A thirty-second period is advantageously utilized. Of course, the period may be less than one second or more than thirty seconds, depending on the speed of the internal network **102** and the firewalls **102**, and the network conditions on the internal network **102**. It is also contemplated that the period between transmission of CCB update packets **400** need not be fixed, but may vary depending on network conditions and on network traffic through the firewalls **100** between the internal network **102** and the external network **104**. It is also within the scope of the preferred embodiment to issue a CCB update packet **400** from a firewall **100** only after the firewall **100** makes a new TCP connection, thus allowing the firewall **100** to provide highly current network state data to its peers.

If the firewall **100** pulls one or more CCBs **300** from a network peer, the firewall **100** transmits a request for a CCB update packet **400** to one or more other firewalls **100** on the internal network. The one or more firewalls receiving the request for a CCB update packet **400** each make a copy of the one or more CCBs **300** associated with its TCP connections and transmit those CCBs **300** to the requesting firewall **100** in a CCB update packet **400**. Preferably, the requesting firewall **100** may requests CCB update packets **400** from all of its peers on the internal network. However, it is within the scope of the preferred embodiment for the requesting firewall **100** to request CCB update packets **400** from only a subset of the other firewalls **100**. If only a subset of the other firewalls **100** transmit CCB update packets to the requesting firewall **100**, then it will take longer for CCBs **300** to propagate across the network; indeed, some CCBs **300** may time out before spreading to all of the firewalls **100**. In a preferred embodiment, a firewall **100** pulls a CCB update packet **400** from one or more of its network peers on a periodic basis; that period is preferably fixed and preferably chosen to be within the range of one second to thirty seconds. A thirty-second period is advantageously utilized. Of course, the period may be less than one second or more than thirty seconds, depending on the speed of the internal network **102** and the firewalls **102**, and the network conditions on the internal network **102**. It is also contemplated that the period between requests for CCB update packets **400** need not be fixed, but may vary depending on network conditions and on network traffic through the firewalls **100** between the internal network **102** and the external network **104**. It is also within the scope of the preferred embodiment to request a CCB update packet **400** from another firewall **100** only before the requesting firewall **100** attempts to make a new TCP connection. It is within the scope of the preferred embodiment to switch among different means of sharing CCBs **300** over a period of time, depending on network conditions, traffic volume, number of TCP connections and other factors.

The CCB update packet **400** is preferably transmitted between firewalls **100** using the UDP standard. User Datagram Protocol (UDP) is a simpler transmission protocol than TCP, and is also well-known in the art. Unlike TCP, UDP does not divide a message into packets at the transmitting end or reassemble those packets at the receiving end. Thus, an application utilizing UDP must either itself disassemble, reassemble and check the packets, or alternately not break the transmitted data into packets at all. If small data units are exchanged, UDP may save processing time. The CCB update packet **400** preferably does not need to be broken up into smaller packets, so the use of UDP, which does not itself break data up into packets, provides an efficient low-overhead protocol for the transmission of CCB update packets **400**.

When a firewall **100** receives a CCB update packet **400** from another firewall **100**, the receiving firewall **100** stores the CCBs **300** that were contained in that CCB update packet **400**. In the event that one of the clients **104** requests through a firewall **100** a connection to a particular server **114**, and the firewall **100** has received and stored a CCB **300** for the connection between the firewall **100** and the particular server **114**, the firewall **100** uses that CCB **300** to initiate the TCP connection with the particular server **114**. If the CCB update packet **400** includes one or more CCBs **300** that are already present in the firewall **100**, then the firewall **100** preferably ignores the duplicate CCBs **300**. In a preferred embodiment, the firewall **100** retains the most recent CCB update packet **400** that it has received from each network peer that has transmitted to it a CCB update packet **400**. When the firewall **100** begins originating a CCB update packet **400** for transmission to a network peer, the firewall **100** preferably checks the most-recently received CCB update packet **400** from that peer. If one of the CCBs **300** in that retained CCB update packet **400** has the same destination server **114** as a CCB **300** in use at the firewall **100**, then that CCB **300** is not placed in the outgoing CCB update packet **400** transmitted to that network peer. In this way, transmission of duplicate CCBs **300** among network peers is reduced. The lifetime of an entry on the duplicate CCB list in a given firewall **100** is preferably substantially equivalent to the lifetime of a CCB **300** itself, thereby allowing the firewalls **100** to adapt to changing network conditions. In a preferred embodiment, the lifetime of a CCB **300** is substantially equal to the maximum segment lifetime (MSL) on the network. In the TCP standard, the MSL is two minutes.

Referring to FIG. **5**, a process is shown for using a CCB **300** when opening a connection from a firewall **100** to an external network **110**. In step **500**, the firewall **100** receives an instruction from a particular client **112** to retrieve data, such as a web page, from a specific external server **114**. The firewall **100** determines if a CCB **300** already exists that contains shared connection state data **306** for a TCP connection between the firewall **100** and the specific server **114**. The presence or absence of such a CCB **300** will affect the actions taken by the firewall **100** to connect to the server **114**. Next, in step **502**, the firewall **100** sends a SYN flag to the specific server **114**. SYN is a common abbreviation for synchronize, and is a standard TCP command for initiating a connection between a client and a server. In the next step **504**, the firewall **100** determines whether a SYN and an ACK have been received back from the specific server **114**. ACK stands for acknowledge, and is a flag indicating that a SYN flag sent from a first computer has been received by a second computer. The SYN flag that is sent from the server **114** back to the firewall **100** represents the attempt by the server **114** to open a connection back to the firewall **100**. The response of SYN and ACK to a SYN flag is part of the TCP standard, and is well known to those skilled in the art. In step **506**, if the particular server **114** has returned a SYN and ACK to the firewall **100**, the firewall **100** updates the CCB **300** with the state information associated with the state of the connection between the firewall **100** and the server **114** as determined from the packet or packets containing the SYN and ACK flags. The firewall **100** then transmits an ACK to the server **114**, as expected in the TCP standard, to open the other end of the connection.

However, if no SYN and ACK are received from the server **114** in step **504**, the process proceeds to step **508**. In step **508**, the firewall **100** determines whether an RST flag has been received from the particular server **114**. RST is an abbreviation for reset, and is a standard TCP command issued from a server to a client after receiving a SYN flag when the server detects a half-open connection or other anomaly. The RST flag is a well-known part of the TCP standard, and its use is well known to those skilled in the art. If the firewall **100** has received an RST flag from the particular server **114**, then the process proceeds to step **510**, where the firewall updates the CCB **300** with the state information associated with the state of the connection between the firewall **100** and the server **114**, as determined from the packet or packets containing the RST flag. The firewall **100** then considers the connection attempt rejected. However, in step **508**, if no RST flag is received from the particular server **114**, the firewall **100** continues to wait for a fixed period of time for a response from the server **114**. This fixed period of time is preferably substantially equal to the round trip time (RTT) estimate. Preferably, if a CCB **300** exists for the connection, then the RTT contained within the CCB is used. If no response is received from the server **114** after that fixed period of time, retransmission preferably may be attempted up to twelve times, using exponentially increasing but bounded waiting periods. For example, the waiting periods may be two times the RTT, then four times the RTT, then eight times the RTT, and so on, up to the upper bound. If no connection is established, the client **112** is notified by the firewall **100** that the connection attempt with the particular server **114** was unsuccessful. No connection is established. By storing state in a CCB **300**, the firewall **100** can declare a server **114** dead if repeated connection requests go unanswered. Further connection requests to the server **114** are preferably rejected by the firewall **100**, with periodic exceptions to test whether the server **114** is no longer dead or unreachable. Situations where declaring a server **114** dead may be useful include attempts to connect to a server **114** experiencing an overwhelming amount of traffic, such as a server **114** operated by a news company that is hosting popular new pictures from a space probe. By blocking attempts to initiate a connection between the firewall **100** and the server **114**, substantial network congestion can be avoided. After step **512**, the process proceeds to step **510**, where the CCB **300** is updated to reflect that a time out occurred in response to a request for data from a particular server **114**. The process from step **510** then proceeds to step **514**, where the connection between the firewall **100** and the server **114** is in a closed state.

Moving back to the main branch of the process after step **506**, the connection between the firewall **100** and the particular server **114** is established in step **516**. This connection is a typical TCP connection in which data is transferred between the server **114** and the firewall **100**. After data has been transmitted from the server **114** to the firewall **100**, the firewall **100** analyzes the data in accordance with its standard policy rules and passes the data along to the specific client **112**. In the prior art, data transfer begins with a slow start. That is, one packet of data is initially transferred. If the packet travels to its destination successfully, two or more packets are transferred. The process continues, ramping up the number of packets constituting each transmission until congestion or other limits affect packet transmission. However, in a preferred embodiment, the CCB **300** already contains state data regarding congestion, RTT and other state components affecting packet transmission. Thus, in a preferred embodiment, the connection between the firewall **100** and a particular server **114** is not initiated with a slow start, but rather by transmitting as many packets as the CCB **300** indicates may be safely sent without substantial congestion. In this way, data transfer is speeded and throughput between the firewall **100** and a particular server **114** is increased. Throughput over the internal network **102** is improved as well, because each TCP connection made in this manner consumes less time than an analogous prior art connection transmitting the same amount of data. The CCB **300** is preferably updated throughout the data transfer process, because the state of the TCP connection between the firewall **100** and the server **144** is subject to change throughout the data transfer process. Preferably, the CCB **300** is updated whenever the round trip time is sampled.

While terminology referring to a specific client **112** and a particular server **114** is used, it is well understood by those skilled in the art that in the course of transmitting data between these two entities, the client **112** may act as a server at some times and the server **114** may act as a client at some times. For example, if the particular client **112** is requesting a search engine web page from a particular server **114**, that web page will be served back to the client **112** by the server **114**. When a user types in a search request into the search engine and transmits that back to the server **114**, the client **112** is temporarily acting as a server and serving data back to the server **114**, which temporarily acts as a client. Indeed, it will be apparent that any data transfer process where there is a back and forth transmission of data, as opposed to a simple one-way stream, will create a situation where each connected computer acts at times as a server and at other times as a client.

After data transfer between the firewall **100** and the server **114** is complete, the process then moves to step **520**, where the connection between the firewall **100** and the server **114** is shut down. This may occur in several ways. Preferably, the firewall **100** transmits a FIN flag to the specific server **114**, or receives a FIN flag from the specific server **114**. FIN is an abbreviation of finish, and is a TCP command that indicates that no more data is left to be transmitted from the sender. The use of the FIN flag is preferred because it allows for quick closure of connections that are no longer needed, and the consequent freeing up of valuable network resources. However, under some circumstances the connection between the firewall **100** and the particular server **114** may be shut down by a time out, which may occur when there is outstanding data to be acknowledged and the sender does not receive a response after roughly one RTT period. A series of retransmissions with exponentially increasing but bounded waiting periods are then attempted, before the connection is deemed dead and shut down by the firewall **100**. Of course, the firewall **100** and the server **114** may maintain a connection after data transmission is complete; whether the connection is maintained or closed after data transmission is finished is based on the application that requested the connection, not the operating system or TCP. For example, Telnet sessions may remain quiescent for hours, days or even longer before resuming activity, if neither the source nor the destination hosts were rebooted. However, even for a long-duration connection having quiescent periods, shutdown is performed as disclosed above when the application that requested the connection closes it, or when the connection is prematurely terminated due to network problems or other errors.

In step **522**, the firewall **100** updates the CCB **300** associated with the connection between the firewall **100** and the particular server **114**, based on the results of the shutdown of the connection. After the CCB **300** is updated in step **522** to reflect the results of the shutdown as a component of the connection state, the connection between the firewall **100** and the particular server **114** is closed. In a preferred embodiment, the CCB **300** associated with this particular connection between the firewall **100** and the particular server **114** is retained for a duration substantially equal to the maximum segment lifetime on the network. In the TCP standard, that duration is approximately two minutes. Preferably, that CCB **300** is then deleted if it has not been used during that time.

FIG. **6** shows a process for utilizing a CCB **300** when receiving a request for connection at the firewall **100** from an external network **110**. In step **600**, the firewall **100** receives a SYN flag from a particular server **114**. Moving to step **602**, the firewall **100** updates the CCB **300** associated with the state information associated with the state of the connection between the firewall **100** and the server **114**, as determined from the packet or packets containing the SYN flag. Moving to step **604**, the firewall **100** then sends SYN and ACK flags to the particular server **114**. The ACK flag acknowledges the receipt of the SYN flag from the server **114** and the transmission of a SYN flag initiates a connection from the firewall **100** to the particular server **114**. Of course, if there is a problem, the firewall **100** may send an RST flag to the server **114** instead; the server **114** will then consider the connection attempt rejected. Moving to step **606**, the firewall **100** determines whether it has received an ACK flag from the particular server **114** in response to the SYN flag sent to it. If the firewall **100** has received an ACK flag from the server **114**, the firewall **100** updates the CCB **300** in step **608**. The connection between the firewall **100** and the server **114** is established when the firewall **100** receives the ACK flag. However, if no ACK flag is received from the particular server **114** in step **606**, then in step **610** the server determines whether it has received an RST flag from the particular server **114**. If so, the process moves to step **612**, where the CCB **300** associated with the connection is updated to reflect the receipt of the RST flag. If in step **610** no RST flag has been received by the firewall **100**, then the firewall **100** waits until the connection times out in step **614**. The timing out process of step **614** is preferably the same as disclosed above in regard to step **512** of the process shown in FIG. **5**. If the connection between the firewall **100** and the particular server **114** times out, the CCB **300** is updated in step **612** to reflect that. After step **612**, the process moves to step **616** in which the connection between the firewall **100** and the server **114** is in a closed state.

Where an ACK flag was received from the server in step **606** and the CCB was updated in step **608**, a connection is established between the firewall **100** and the particular server **114** in step **618**, and data transfer occurs. The disclosure above, regarding the establishment of a connection and the transfer of data between an individual client **112** and the particular server **114** in step **516** of the process shown in FIG. **5**, applies to step **618** as well. Moving to step **620**, the CCB **300** is preferably updated with the connection state throughout the data transfer process, because the state of the TCP connection between the firewall **100** and the server **144** is subject to change throughout the data transfer process. Preferably, the CCB **300** is updated whenever the round trip time is sampled.

Moving to step **622**, after data transfer between the particular client **114** and the particular server **114** is concluded, the connection is shutdown. This shutdown may occur by sending or receiving a FIN flag or by waiting for the connection to time out as described above in regard to step **520** of the process shown in FIG. **5**. The process then moves to step **624**, where the CCB **300** is updated with the results of the shutdown in step **622**. That is, the type of connection shutdown is reflected in the CCB **300** as a component of the state of the connection. Moving to step **626**, the connection between the firewall **100** and the particular server **114** is then closed. In a preferred embodiment, the CCB **300** associated with this particular connection between the firewall **100** and the particular server **114** is retained for a duration substantially equal to the network MSL. In the case of the TCP standard, that duration is approximately two minutes. Preferably, that CCB **300** is then deleted if it has not been used during that time.

In a preferred embodiment, each firewall **100** adjusts data traffic passing through it based on the contents of the CCBs **300** stored within it. The CCBs **300** provide an individual firewall **100** with data regarding the overall conditions on the internal network **102**. The policy decisions made by the firewall **100** are based on concerns including security, bandwidth, and resource utilization. By utilizing data regarding the state of the internal network **102**, the firewall **100** can make better policy decisions that allow for enhanced throughput between the internal network **102** and the external network **110**. In a preferred embodiment, each firewall **100** periodically reviews the CCBs **300** stored within it to determine of the overall state of the internal network **102**. Preferably, that individual firewall **100** then adjusts the data traffic through it, based on the data traffic through its network peers.

Each individual firewall **100** preferably adjusts the data rate through itself to minimize or prevent network saturation. Network saturation is a problem, known to those skilled in the art, that can result in data loss or in reduced throughput between two networks. Network saturation occurs when the firewalls **100** attempt to transmit more data packets between the internal network **102** and the external network **110** than available bandwidth can support. For example, saturation occurs when the internal network **102** is capable of receiving X packets per second from the external network **110** but the external network is transmitting X+1 packets per second or more to the internal network **102**. If an individual firewall **100** is aware of the data transmission rates for the connections originating at that firewall **100**, as well as the data rates and shared connection state data **306** for its network peers, that firewall **110** can adjust the rate of the connections through itself to prevent saturation. By sharing CCBs **300** on a relatively frequent basis, preferably at intervals of one to thirty seconds, each firewall **100** obtains a substantially current picture of the overall state of the internal network **102**. If one firewall **100** suddenly requires additional band width to handle data requests through it, its network peers preferably reduce the data traffic rate through themselves to prevent network saturation. Those network peers have the knowledge to reduce that data traffic rate based on the shared connection state data **306** within the CCBs **300** they have received from the firewall **100** requiring additional bandwidth. Because the CCBs **300** provide knowledge to a firewall **100** of the behavior of its network peers, throughput can be improved by allowing each firewall **100** to better adjust the traffic through itself to reduce or prevent network saturation.

A preferred system for sharing network state to enhance throughput, and many of its attendant advantages, has thus been disclosed. It will be apparent, however, that various changes may be made in the content and arrangement of the steps of the process without departing from the spirit and scope of the invention, the method hereinbefore described being merely preferred or exemplary embodiments thereof. Therefore, the invention is not to be restricted or limited except in accordance with the following claims and their legal equivalents.

What is claimed is:

1. A method for enhancing network throughput between an internal network and an external network to which a server is connected, comprising the steps of:

connecting two or more firewalls to the internal network;

determining whether a common TCP control block exists for a TCP connection between one of said firewalls and the server, and creating one if one does not exist;

sending a TCP connection request to the server from one of said firewalls; and

updating said common TCP control block based on the response from the server to said TCP connection request;

wherein said steps further comprise establishing a connection between said firewall and said server, and updating said common TCP control block with connection state data during said connection;

wherein said steps further comprise shutting down said connection, and updating said common TCP control block based on the type of shutdown performed;

wherein said common TCP control block is shared with one or more of said other firewalls.

2. The method of claim **1**, wherein the external network includes the Internet.

3. A method for enhancing network throughput between an internal network and an external network to which a server is connected, comprising the steps of:

connecting two or more firewalls to the internal network;

receiving a TCP connection request from the server to one of said firewalls;

determining whether a common TCP control block exists for a TCP connection between said receiving firewall and said server, and creating one if one does not exist; and

updating said common TCP control block based on the TCP connection request from the server;

wherein said steps further comprise transmitting an acknowledgement and a request for connection to the server, and updating said common TCP control block with the resulting connection state data;

wherein said steps further comprise establishing a connection between said firewall and the server and updating said common TCP control block during said connection with connection state data;

wherein said steps further comprise shutting down said connection, and updating said common TCP control block based on the type of shutdown performed;

wherein said common TCP control block is shared with one or more of said other firewalls.

4. The method of claim **3**, wherein the external network includes the Internet.

5. A computer program product embodied on a computer readable medium for enhancing network throughput between an internal network and an external network to which a server is connected, comprising:

computer code for connecting two or more firewalls to the internal network;

computer code for determining whether a common TCP control block exists for a TCP connection between one of said firewalls and the server, and creating one if one does not exist;

computer code for sending a TCP connection request to the server from one of said firewalls; and

computer code for updating said common TCP control block based on the response from the server to said TCP connection request;

wherein a connection is established between said firewall and said server, and said common TCP control block is updated with connection state data during said connection;

wherein said connection is shut down, and said common TCP control block is updated based on the type of shutdown performed;

wherein said common TCP control block is shared with one or more of said other firewalls.

6. The computer program product of claim **5**, wherein the external network includes the Internet.

7. A computer program product embodied on a computer readable medium for enhancing network throughput between an internal network and an external network to which a server is connected, comprising:

computer code for connecting two or more firewalls to the internal network;

computer code for receiving a TCP connection request from the server to one of said firewalls;

computer code for determining whether a common TCP control block exists for a TCP connection between said receiving firewall and said server, and creating one if one does not exist; and

computer code for updating said common TCP control block based on the TCP connection request from the server;

wherein an acknowledgement and a request for connection is sent to the server, and said common TCP control block is updated with the resulting connection state data;

wherein a connection is established between said firewall and the server, and said common TCP control block is updated during said connection with connection state data;

wherein said connection is shut down, and said common TCP control block is updated based on the type of shutdown performed;

wherein said common TCP control block is shared with one or more of said other firewalls.

8. The computer program product of claim **7**, wherein the external network includes the Internet.

9. An apparatus for enhancing network throughput between an internal network and an external network to which a server is connected, comprising:

logic for connecting two or more firewalls to the internal network;

logic for determining whether a common TCP control block exists for a TCP connection between one of said firewalls and the server, and creating one if one does not exist;

logic for sending a TCP connection request to the server from one of said firewalls; and

logic for updating said common TCP control block based on the response from the server to said TCP connection request;

wherein a connection is established between said firewall and said server, and said common TCP control block is updated with connection state data during said connection;

wherein said connection is shut down, and said common TCP control block is updated based on the type of shutdown performed;

wherein said common TCP control block is shared with one or more of said other firewalls.

10. The apparatus of claim **9**, wherein the external network includes the Internet.

11. An apparatus for enhancing network throughput between an internal network and an external network to which a server is connected, comprising:

logic for connecting two or more firewalls to the internal network;

logic for receiving a TCP connection request from the server to one of said firewalls;

logic for determining whether a common TCP control block exists for a TCP connection between said receiving firewall and said server, and creating one if one does not exist; and

logic for updating said common TCP control block based on the TCP connection request from the server;

wherein an acknowledgement and a request for connection is sent to the server, and said common TCP control block is updated with the resulting connection state data;

wherein a connection is established between said firewall and the server, and said common TCP control block is updated during said connection with connection state data;

wherein said connection is shut down, and said common TCP control block is updated based on the type of shutdown performed;

wherein said common TCP control block is shared with one or more of said other firewalls.

12. The apparatus of claim **11**, wherein the external network includes the Internet.

* * * * *

US005852717A

United States Patent [19]

Bhide et al.

[11] **Patent Number: 5,852,717**

[45] **Date of Patent: Dec. 22, 1998**

[54] **PERFORMANCE OPTIMIZATIONS FOR COMPUTER NETWORKS UTILIZING HTTP**

[75] Inventors: **Chandrashekhar W. Bhide**, Sunnyvale; **Jagdeep Singh**; **Don Oestreicher**, both of Cupertino, all of Calif.

[73] Assignee: **Shiva Corporation**, Bedford, Mass.

[21] Appl. No.: **752,500**

[22] Filed: **Nov. 20, 1996**

[51] **Int. Cl.**^{6} **G06F 17/00**

[52] **U.S. Cl.** **395/200.33**; 395/200.49; 395/200.59

[58] **Field of Search** 395/200.32, 200.33, 395/200.58, 200.59, 200.49, 200.5; 364/551.01

[56] **References Cited**

U.S. PATENT DOCUMENTS

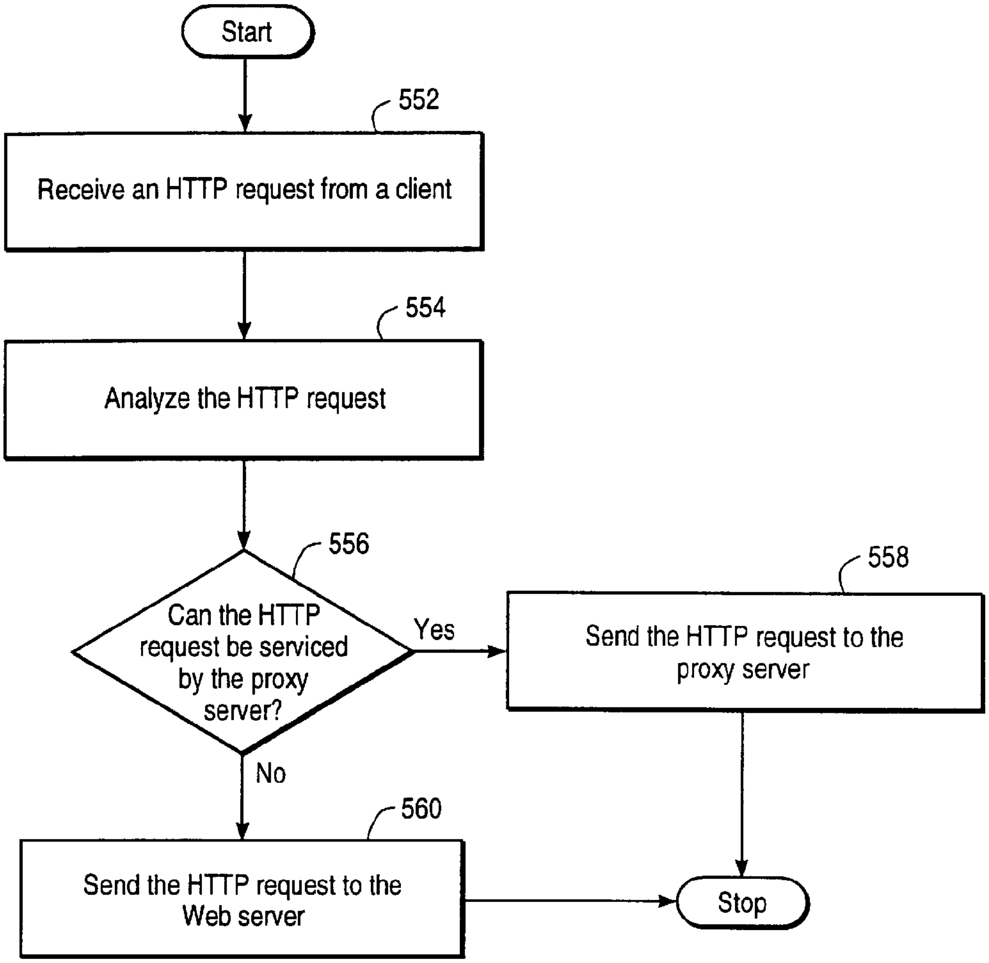
4,891,785	1/1990	Donohoo	395/200.32
5,339,435	8/1994	Lubkin et al.	395/200.5
5,710,918	1/1998	Lagarde et al.	395/200.32

Primary Examiner—Ellis B. Ramirez
Attorney, Agent, or Firm—Townsend and Townsend and Crew LLP

[57] **ABSTRACT**

Systems and methods of increasing the performance of computer networks, especially networks connecting users to the Web, are provided. Performance is increased by reducing the latency the client experiences between sending a request to the server and receiving a response. A connection cache may be maintained by an agent on the network access equipment to more quickly respond to request for network connections to the server. Additionally, the agent may maintain a cache of information to more quickly respond to requests to get an object if it has been modified. These enhancements and other described herein may be implemented singly or in conjunction to reduce the latency involved in sending the requests to the server by saving round-trip times between computer network components.

48 Claims, 15 Drawing Sheets


Start
552
Receive an HTTP request from a client
554
Analyze the HTTP request
556
Can the HTTP request be serviced by the proxy server?
Yes
558
Send the HTTP request to the proxy server
No
560
Send the HTTP request to the Web server
Stop

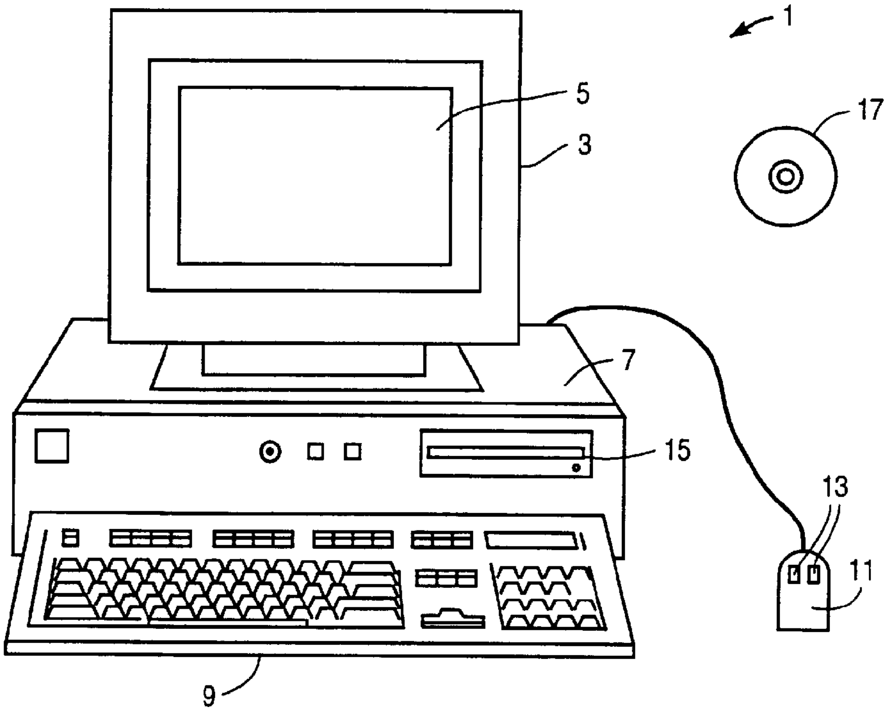

1
5
3
17
7
15
13
11
9

FIG. 1

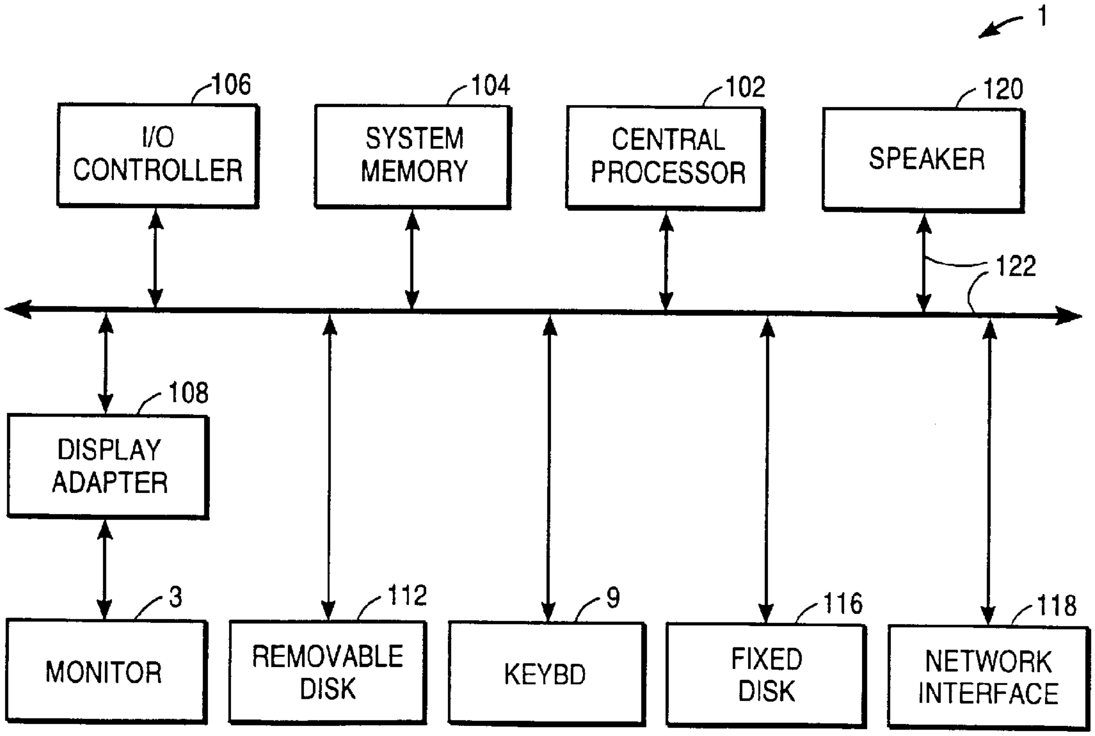

1
106
I/O CONTROLLER
104
SYSTEM MEMORY
102
CENTRAL PROCESSOR
120
SPEAKER
122
108
DISPLAY ADAPTER
3
MONITOR
112
REMOVABLE DISK
9
KEYBD
116
FIXED DISK
118
NETWORK INTERFACE

FIG. 2

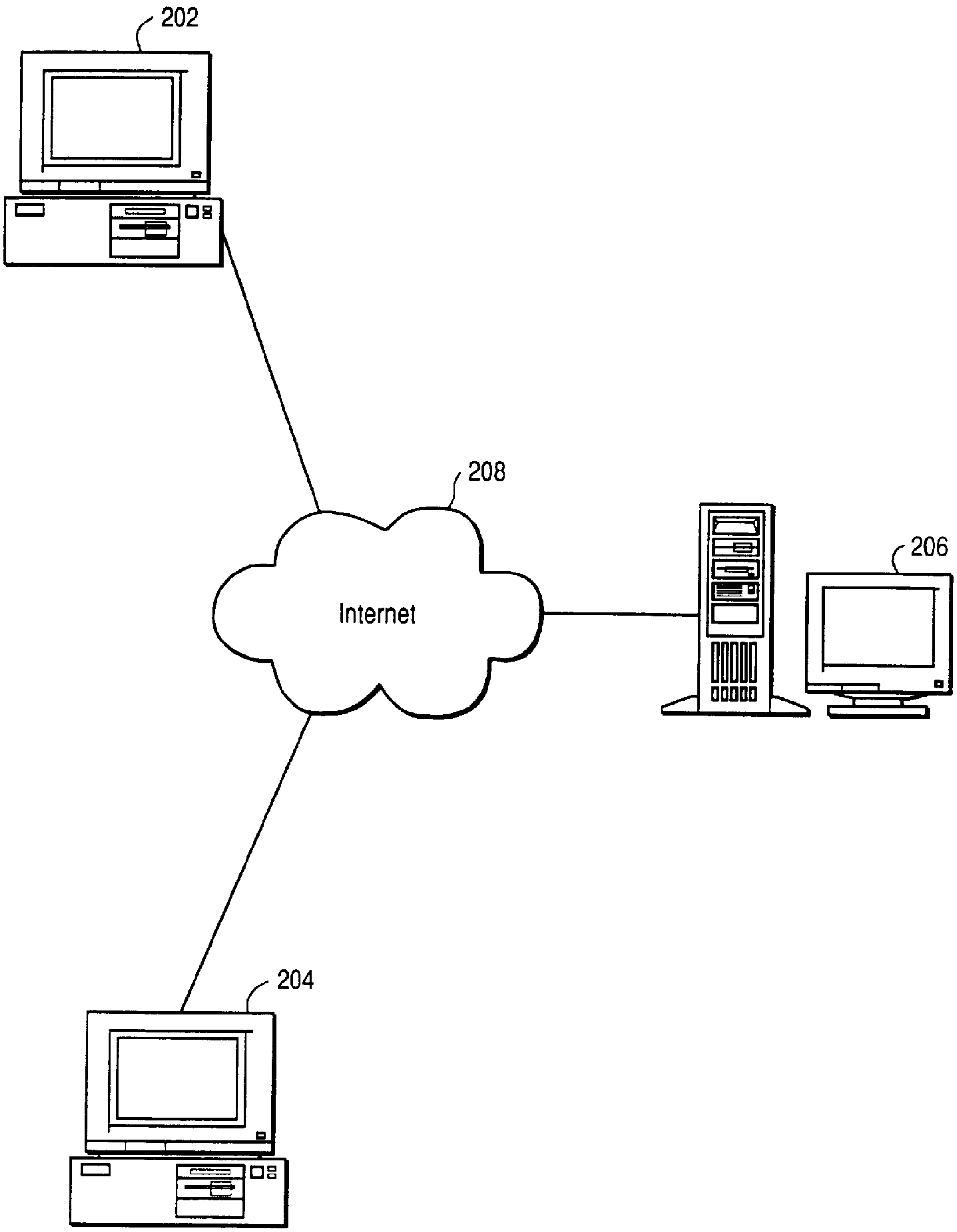


FIG. 3

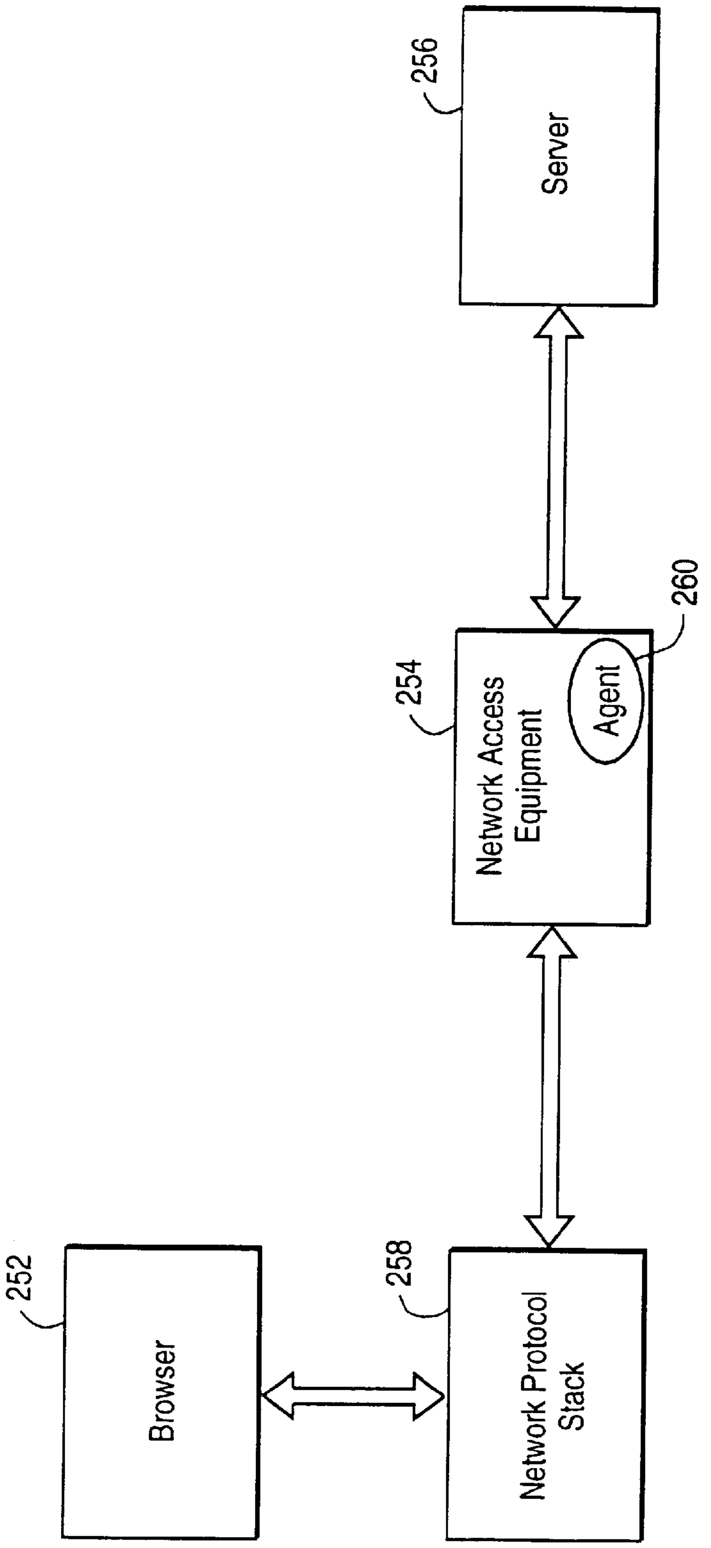

256
Server
254
Network Access Equipment
Agent
260
252
Browser
258
Network Protocol Stack

FIG. 4

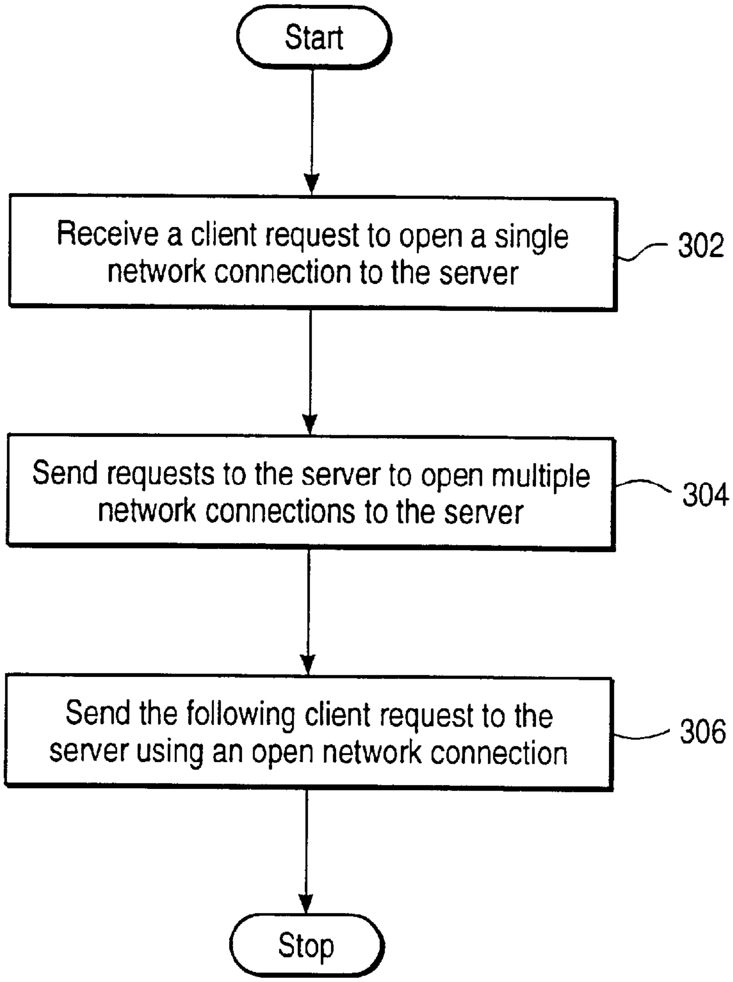

Start
Receive a client request to open a single network connection to the server
302
Send requests to the server to open multiple network connections to the server
304
Send the following client request to the server using an open network connection
306
Stop

FIG. 5

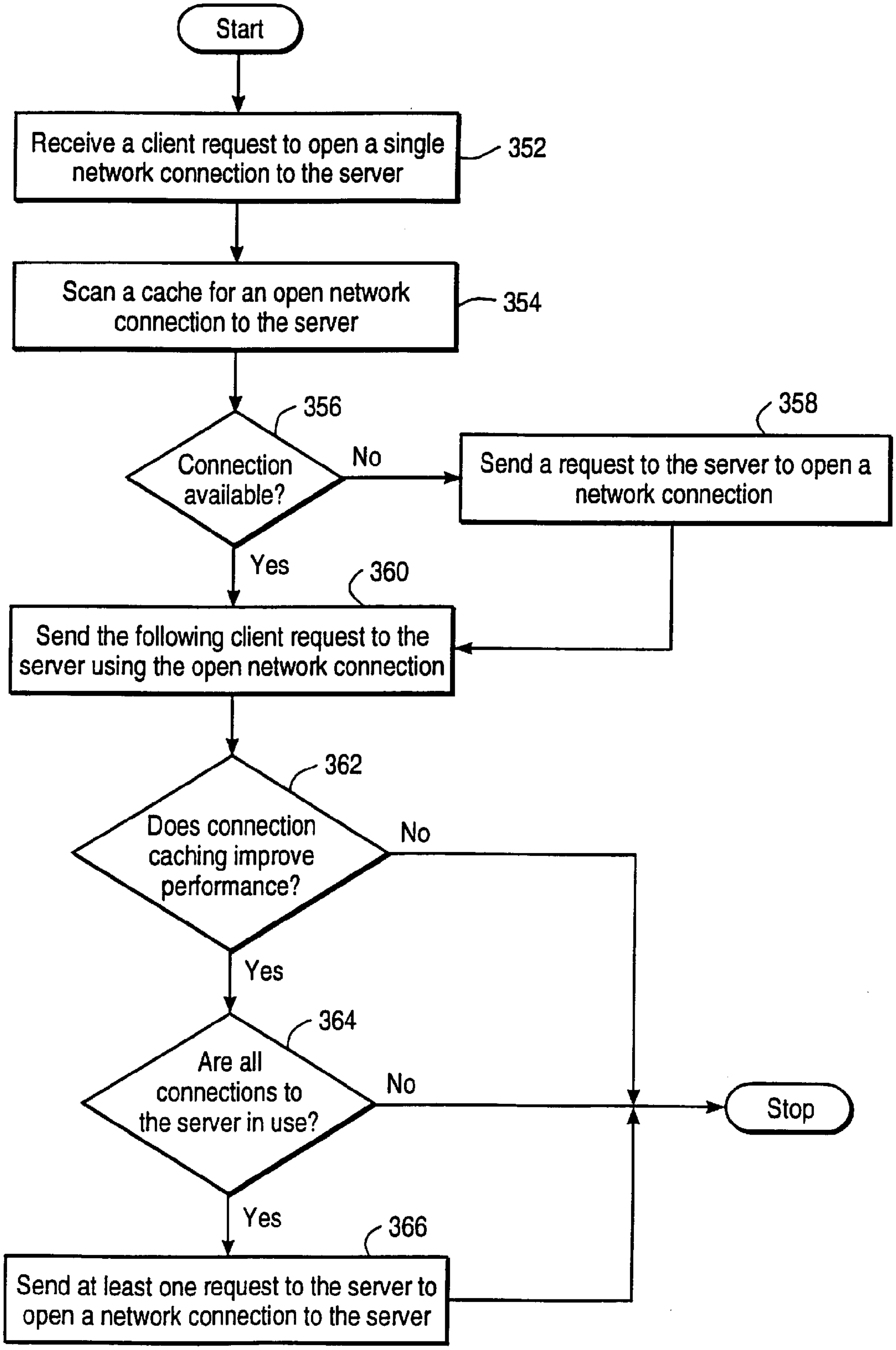

Start
Receive a client request to open a single network connection to the server
352
Scan a cache for an open network connection to the server
354
356
Connection available?
No
358
Send a request to the server to open a network connection
Yes
360
Send the following client request to the server using the open network connection
362
Does connection caching improve performance?
No
Yes
364
Are all connections to the server in use?
No
Stop
Yes
366
Send at least one request to the server to open a network connection to the server

FIG. 6

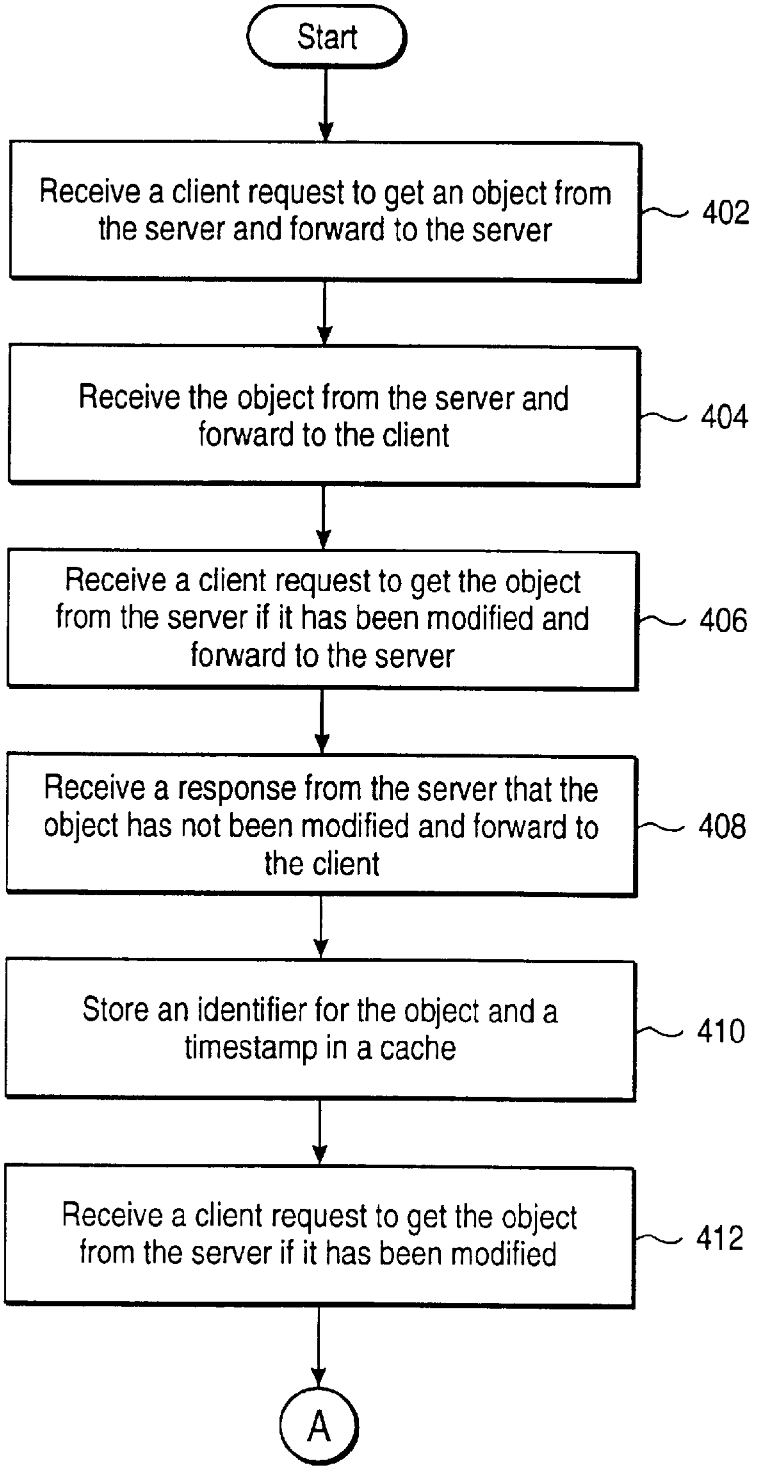

Start
Receive a client request to get an object from the server and forward to the server
402
Receive the object from the server and forward to the client
404
Receive a client request to get the object from the server if it has been modified and forward to the server
406
Receive a response from the server that the object has not been modified and forward to the client
408
Store an identifier for the object and a timestamp in a cache
410
Receive a client request to get the object from the server if it has been modified
412
A

FIG. 7A

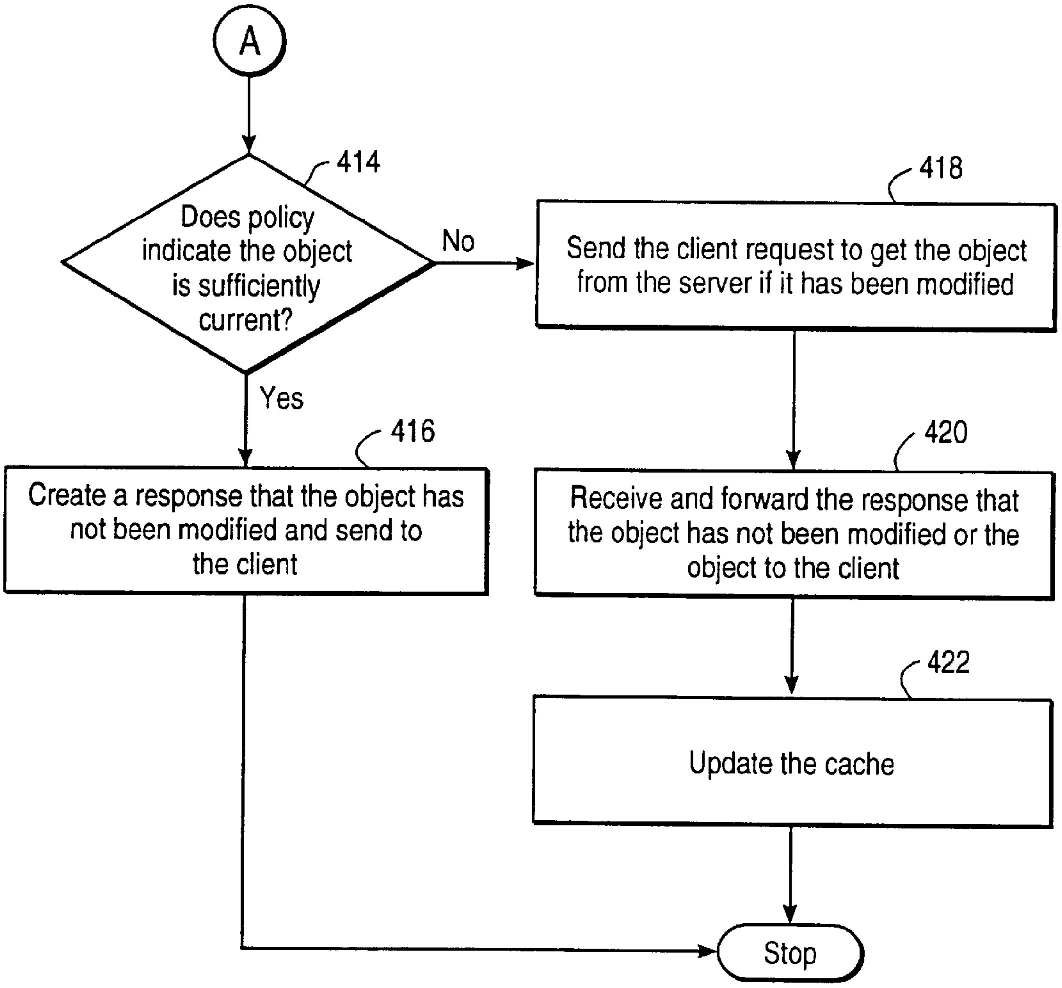

A
414
Does policy indicate the object is sufficiently current?
No
Yes
418
Send the client request to get the object from the server if it has been modified
416
Create a response that the object has not been modified and send to the client
420
Receive and forward the response that the object has not been modified or the object to the client
422
Update the cache
Stop

FIG. 7B

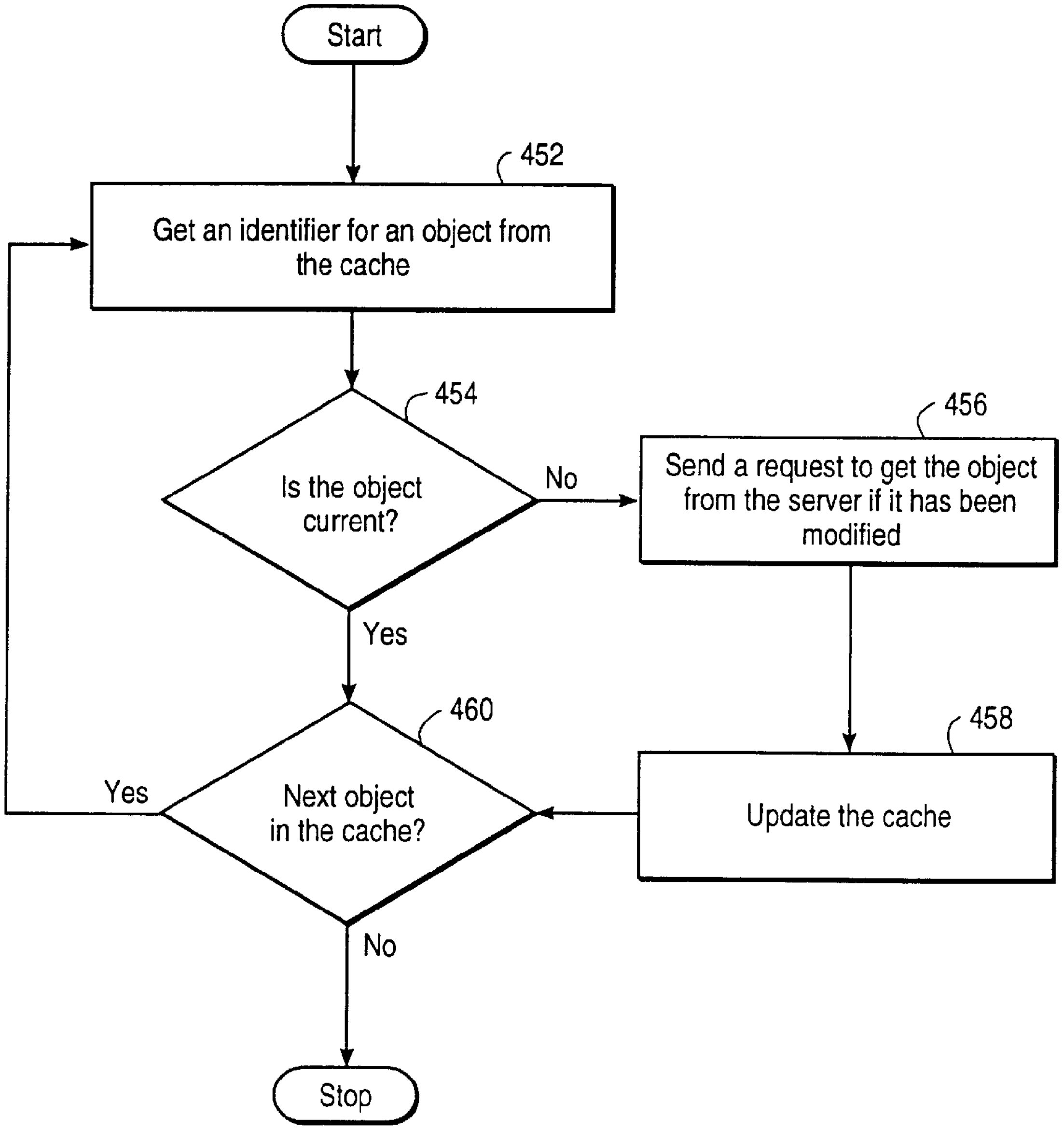

Start
452
Get an identifier for an object from the cache
454
Is the object current?
No
456
Send a request to get the object from the server if it has been modified
Yes
460
458
Yes
Next object in the cache?
Update the cache
No
Stop

FIG. 8

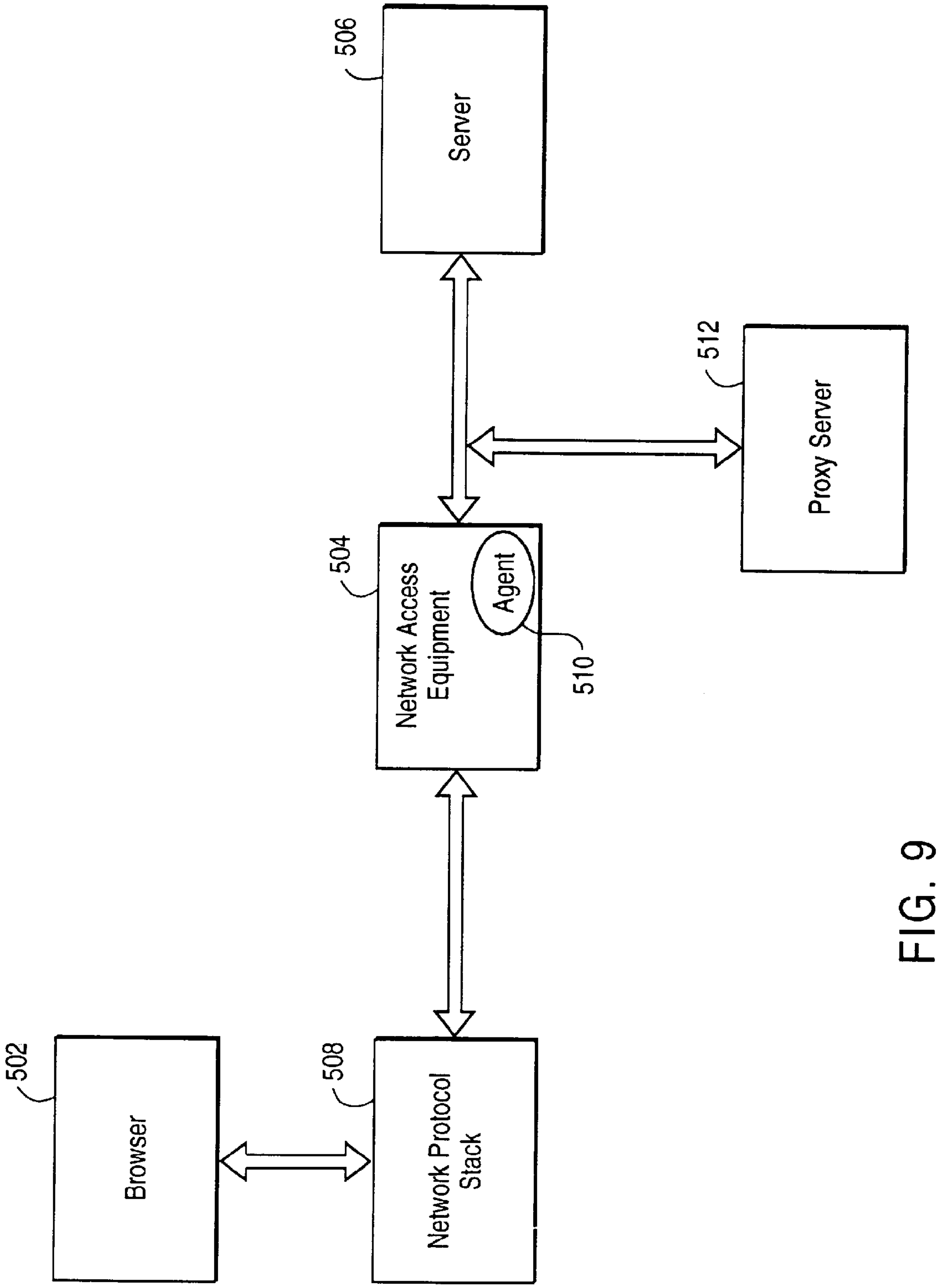

506
Server
512
Proxy Server
504
Network Access Equipment
Agent
510
502
Browser
508
Network Protocol Stack

FIG. 9

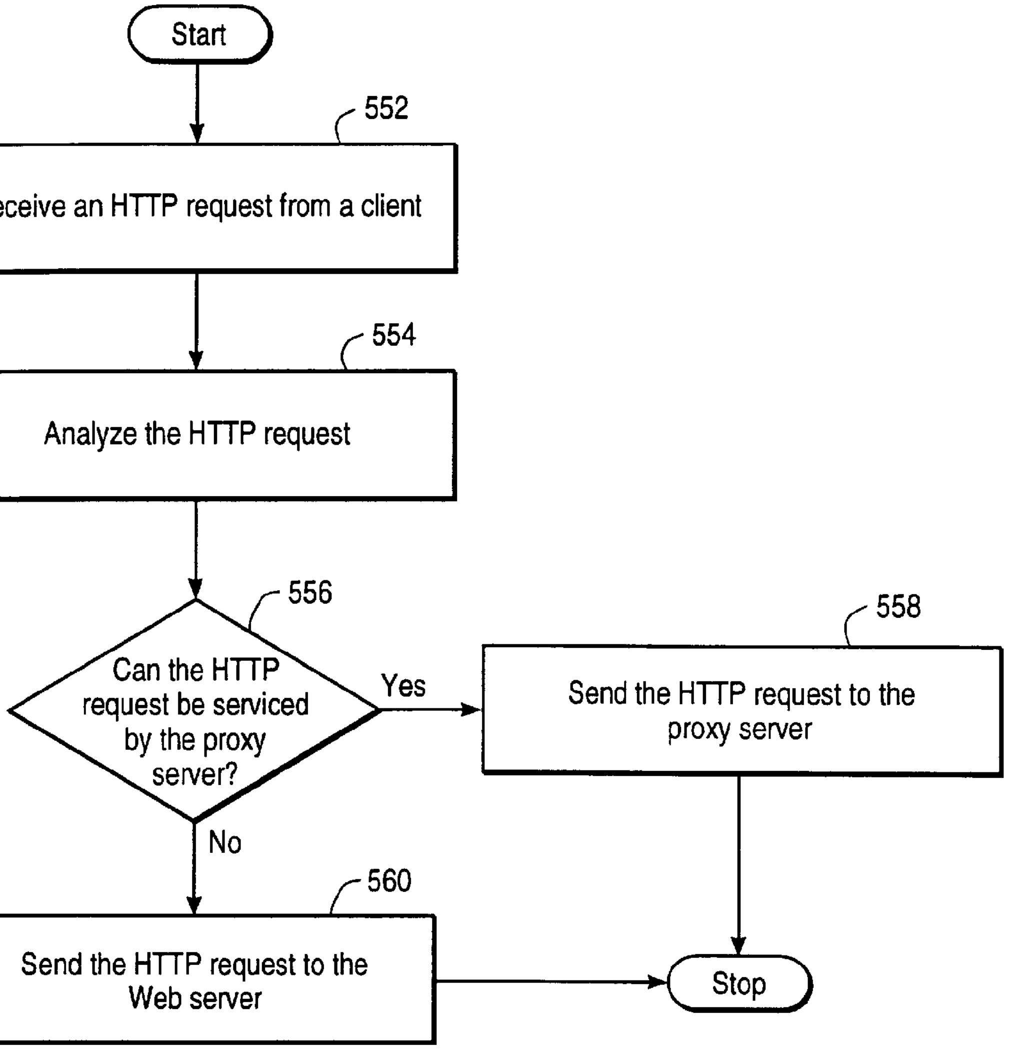

Start
552
Receive an HTTP request from a client
554
Analyze the HTTP request
556
Can the HTTP request be serviced by the proxy server?
Yes
558
Send the HTTP request to the proxy server
No
560
Send the HTTP request to the Web server
Stop

FIG. 10

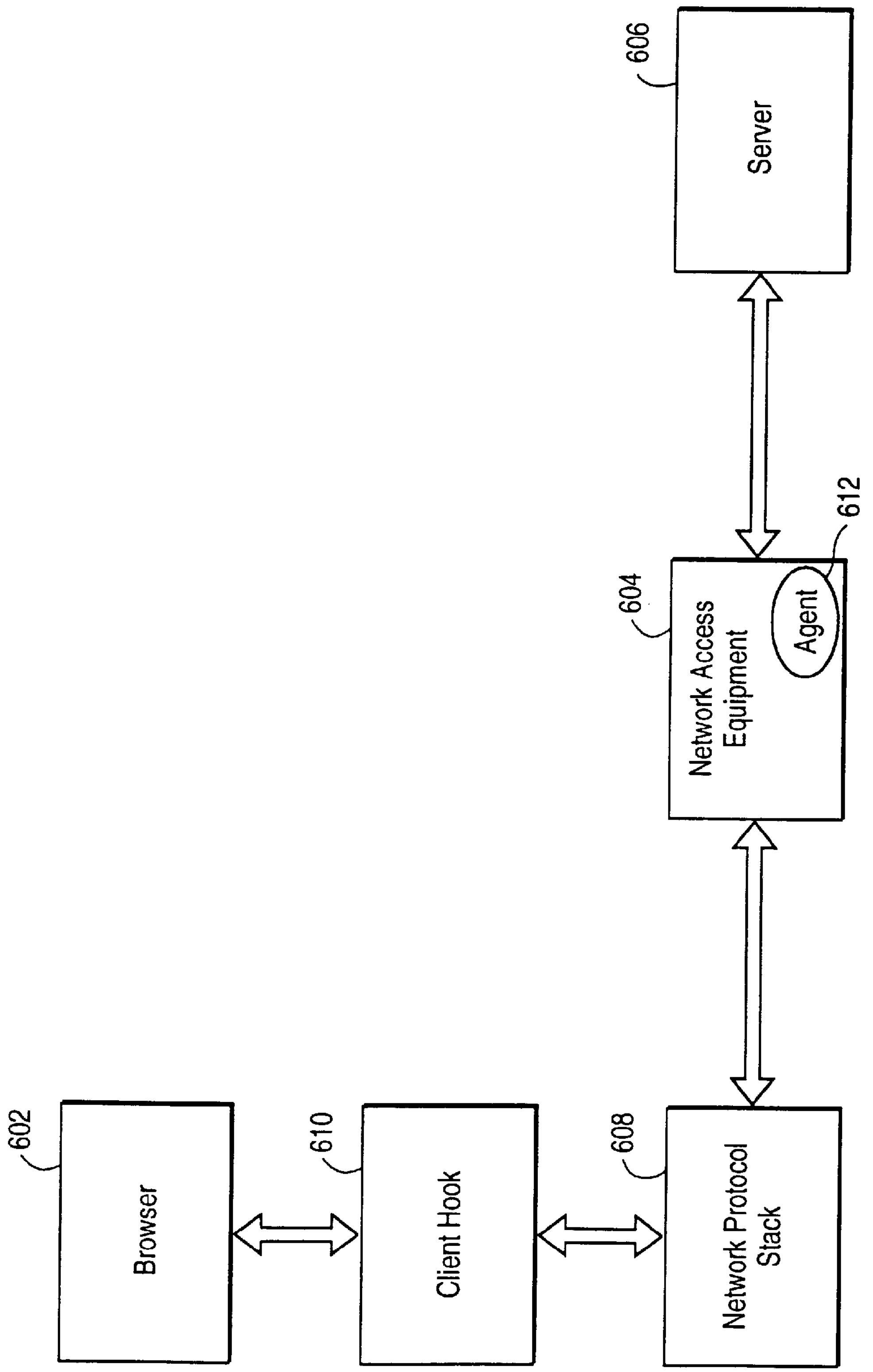

606
Server
604
Network Access Equipment
Agent
612
602
Browser
610
Client Hook
608
Network Protocol Stack

FIG. 11

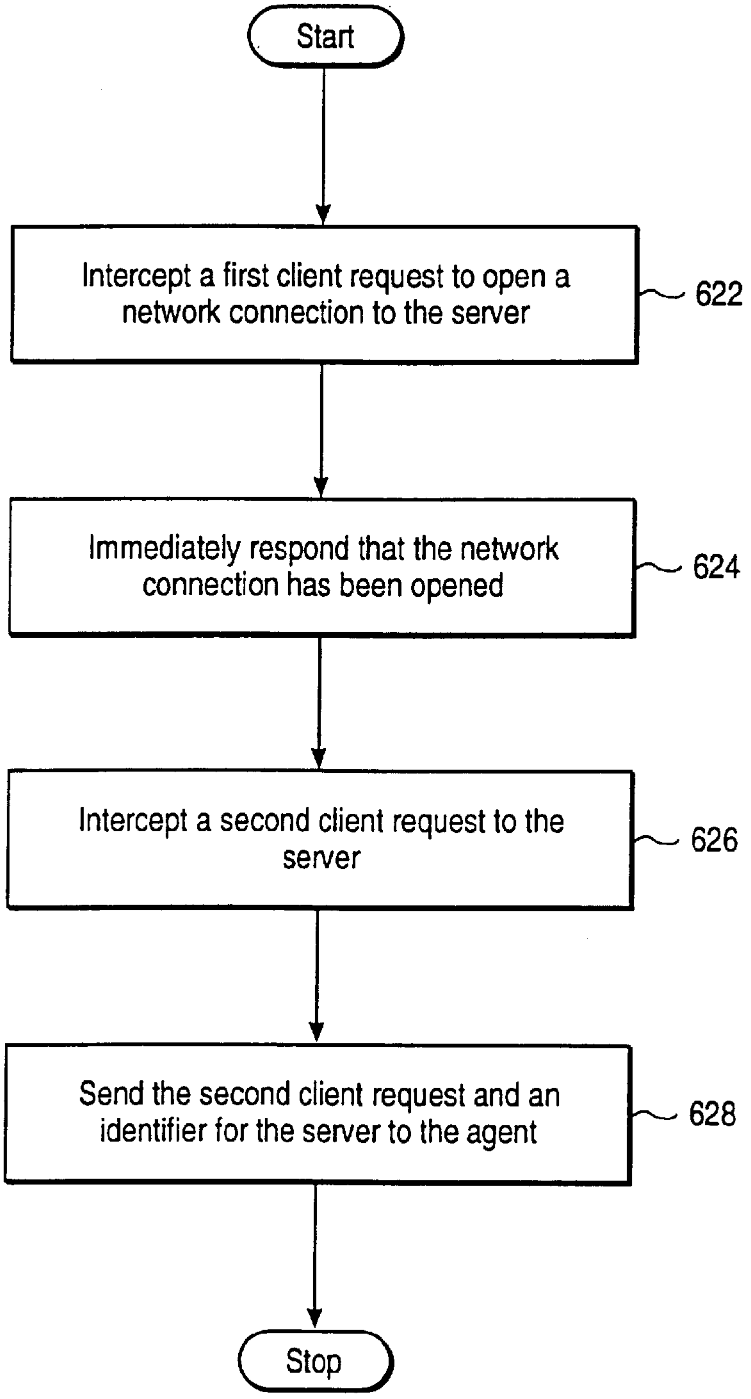

Start
Intercept a first client request to open a network connection to the server
622
Immediately respond that the network connection has been opened
624
Intercept a second client request to the server
626
Send the second client request and an identifier for the server to the agent
628
Stop

FIG. 12

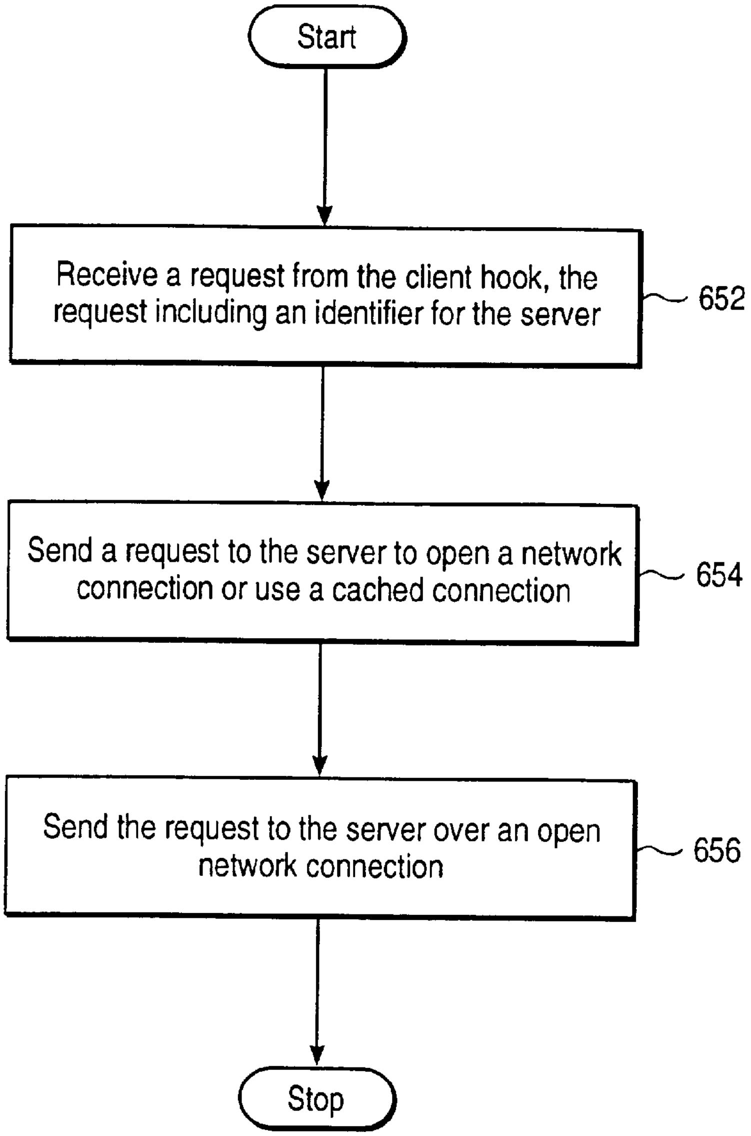

Start
Receive a request from the client hook, the request including an identifier for the server
652
Send a request to the server to open a network connection or use a cached connection
654
Send the request to the server over an open network connection
656
Stop

FIG. 13

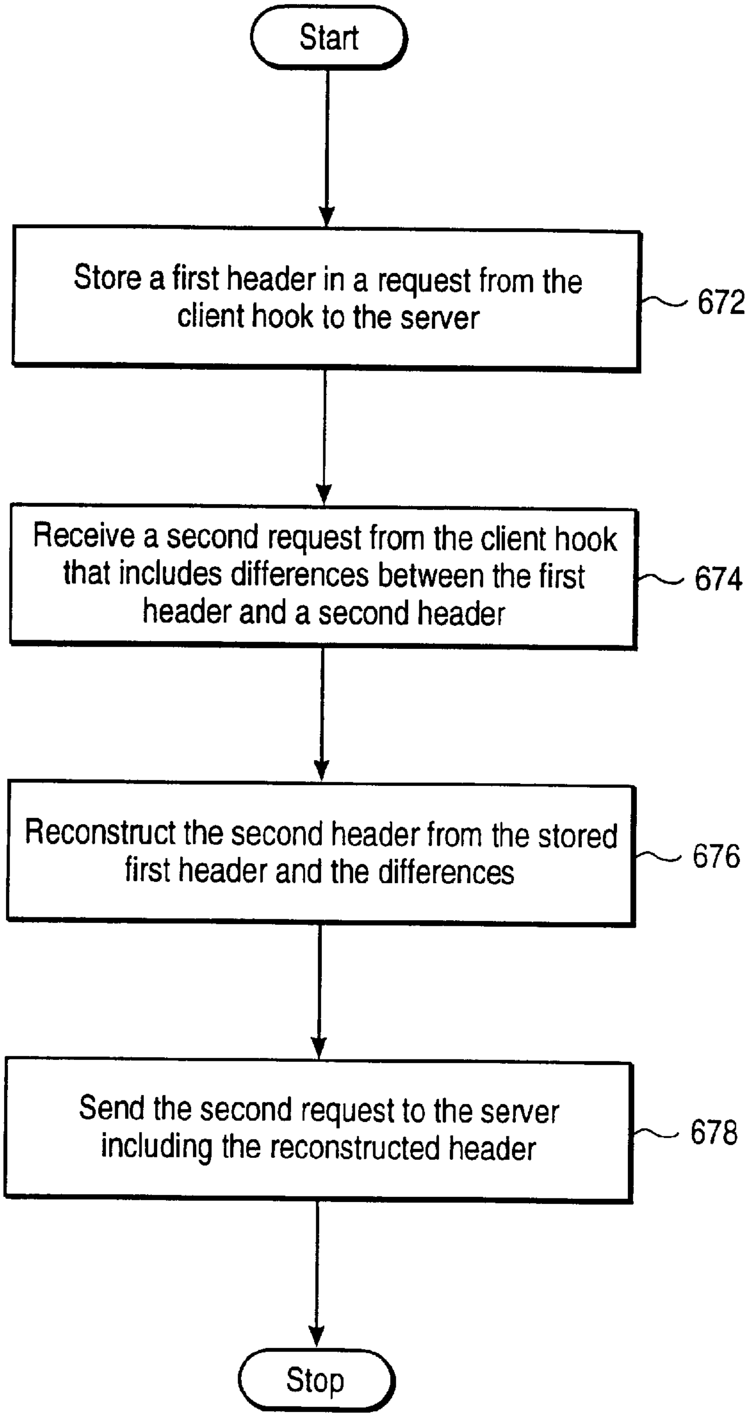

Start
Store a first header in a request from the client hook to the server
672
Receive a second request from the client hook that includes differences between the first header and a second header
674
Reconstruct the second header from the stored first header and the differences
676
Send the second request to the server including the reconstructed header
678
Stop

FIG. 14

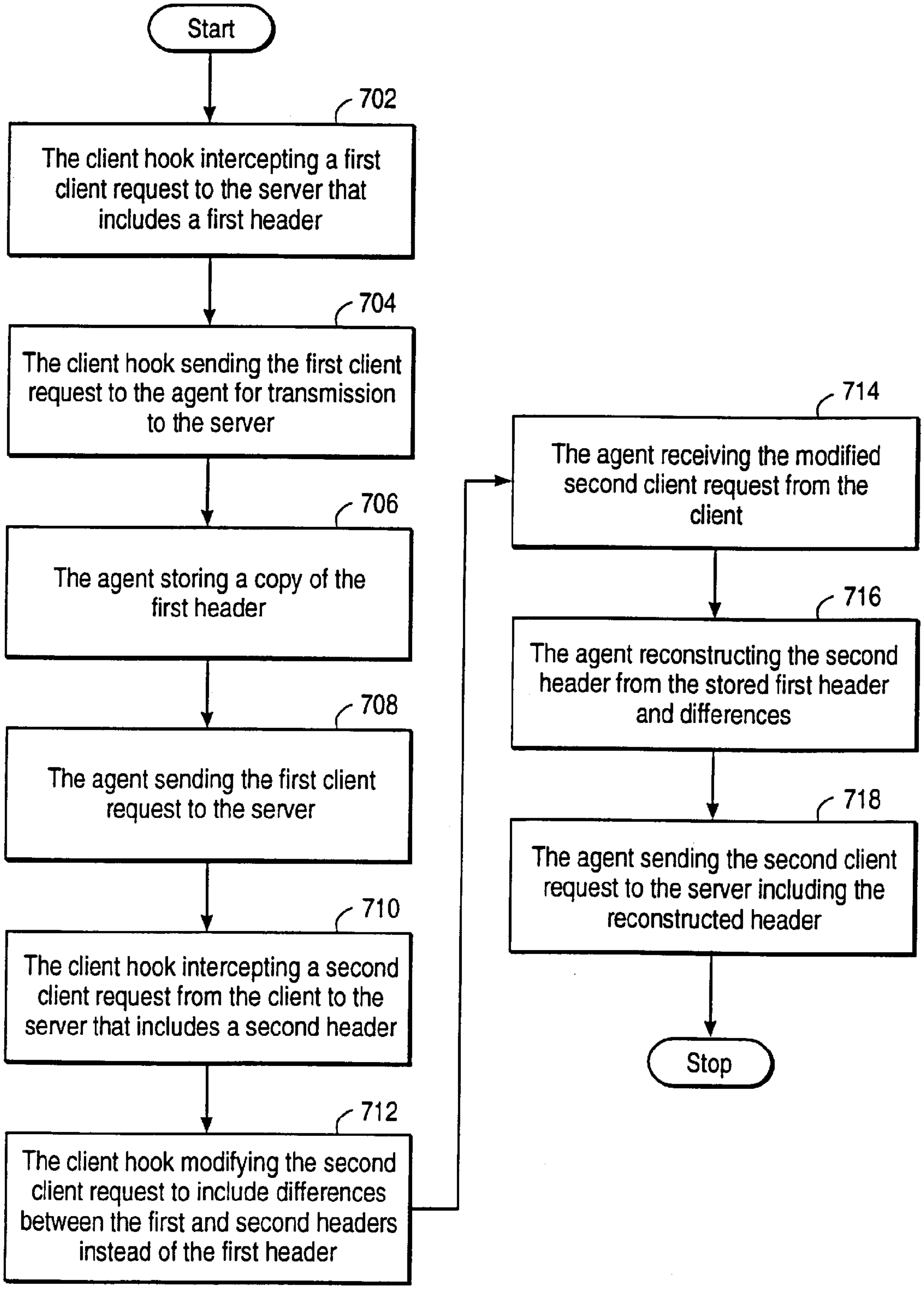

Start
702
The client hook intercepting a first client request to the server that includes a first header
704
The client hook sending the first client request to the agent for transmission to the server
706
The agent storing a copy of the first header
708
The agent sending the first client request to the server
710
The client hook intercepting a second client request from the client to the server that includes a second header
712
The client hook modifying the second client request to include differences between the first and second headers instead of the first header
714
The agent receiving the modified second client request from the client
716
The agent reconstructing the second header from the stored first header and differences
718
The agent sending the second client request to the server including the reconstructed header
Stop

FIG. 15

PERFORMANCE OPTIMIZATIONS FOR COMPUTER NETWORKS UTILIZING HTTP

BACKGROUND OF THE INVENTION

The present invention is related to increasing performance of networked computer systems and, more particularly, increasing the performance of computer systems accessing the World Wide Web ("Web") on the Internet.

The Internet is a network which provides avenues for worldwide communication of information, ideas and messages. Although the Internet has been utilized by academia for decades, recently there has been almost an explosion of interest in the Internet and the information residing thereon. The Web accounts for a significant part of the growth in the popularity of the Internet, perhaps because of the user-friendly graphical user interfaces ("GUIs") of browsers that are readily available for accessing the Web.

The World Wide Web makes hypertext documents available to users over the Internet. A hypertext document does not present information linearly like a book, but instead provides the reader with links or pointers to other locations so that the user may jump from one location to another. The hypertext documents on the Web are accessed through the client/server protocol of Hypertext Transport Protocol ("HTTP").

The Internet utilizes the Transmission Control Protocol/Internet Protocol ("TCP/IP") to network very diverse and dissimilar systems. In Windows 3.x environments, the browser typically utilizes a dynamic link library WINSOCK.DLL to communicate with the TCP/IP-based Internet. Although the hardware backbone of the Internet is a series of high-speed communications links between educational, research, government, and commercial mainframe computer systems, a great number of the users that access the Web utilize a browser that is connected to the Internet through a relatively slow or weak link (e.g., a 28.8K modem over an analog phone line) to network access equipment networked to the Internet.

The network access equipment typically has a fast connection to the Internet (e.g., a T-1 connection at 1.54 MB). Network access equipment may be a remote access server for allowing remote users to connect to intranet and Internet resources. Such a remote access server, the LanRover™ Access Switch remote access server, is available from Shiva Corporation, Bedford, Mass. Other types of network access equipment are utilized by Internet Service Providers ("ISPs") to provide Internet access to customers. Thus, the network access equipment is networked between the computer running the browser and the Web server providing what is called the Point of Presence ("POP") for the user.

Network performance in general is hampered because the network link between users and their POP commonly has a significantly lower bandwidth than the network link between the POP and the Web server. Additionally, there is a significant amount of latency in conventional networks while the client waits for a response from the Web server. Accordingly, there is a need for systems and methods for increasing the performance of the computer networks, preferably without requiring modification of existing browsers.

SUMMARY OF THE INVENTION

The present invention provides systems and methods of increasing the performance of computer networks, especially networks connecting users to the Web. Performance may be increased by reducing the latency the client experiences between sending a request to the server and receiving a response. A connection cache may be maintained by an agent on the network access equipment to more quickly respond to request for network connections to the server. Additionally, the agent may maintain a cache of information to more quickly respond to requests to get an object if it has been modified. These enhancements may be implemented singly or in conjunction to reduce the latency involved in sending the respective requests to the server by saving round-trip times between the agent and the server. The invention complements caching provided by browsers and other components (e.g., proxy servers).

Performance may also be increased by the network access equipment sending an HTTP request to either the Web server or a proxy server based on an analysis of the HTTP request. The Web browsers may then utilize a proxy server transparently, without specifically sending requests to the proxy server.

Additionally, performance may be increased by effectively increasing the effectively bandwidth of the weak link between the client and the network access equipment. A client hook intercepts client requests to the server and modifies the client requests to increase performance. The modified requests are then sent to the agent which reconstructs the client requests from the modified requests and sends the client requests to the server. For example, multiple client requests may be combined into a single modified requests or individual client requests may be intelligently compressed for more efficient utilization of the weak link.

In one embodiment, the present invention provides a method executed by an agent in a computer network between clients and a server for increasing performance between the clients and the server, the method comprising the steps of: receiving a first request from a client to open a single network connection to the server; sending a plurality of requests to the server to open a plurality of network connections to the server; receiving a second request from the client; and sending the second request to the server using one of the plurality of network connections. Accordingly, the plurality of network connections to the server are opened in response to the first request from the client to open a single network connection.

In another embodiment, the present invention provides a method executed by an agent in a computer network between clients and a server for increasing performance between the clients and the server, the method comprising the steps of: receiving a first request from a client to get an object from the server if the object has been modified after a specific timestamp; sending the first request to the server; receiving a first response from the server that the object has not been modified after the specific timestamp; sending the first response to the client; storing an identifier for the object and a timestamp in a cache; receiving a second request from the client to get the object from the server if the object has been modified after the specific timestamp; and if the timestamp stored in the cache is within a predetermined amount of time from the current time, sending a second response to the client that the object has not been modified after the specific timestamp without sending the second request to the server.

In another embodiment, the present invention provides a method executed by an agent in a computer network between a client and a Web and proxy servers for increasing performance between the client and the Web server, comprising the steps of: receiving an HTTP request from a client; and sending the HTTP request to either the Web server or the proxy server depending on the HTTP request, the proxy server storing information available on the Web server. Accordingly, the client does not need to be modified or configured to utilize the proxy server.

In another embodiment, the present invention provides a method for increasing performance between a client on a client computer and a server utilizing a client hook on the client computer and an agent between the client computer and the server, comprising the steps of: the client hook intercepting requests from the client to the server; the client hook modifying the requests from the client; the client hook sending the modified requests to the agent; the agent reconstructing the requests from the client according to the modified requests; and the agent sending the requests from the client to the server. The client hook may intercept requests from the client to open a network connection to the server and immediately respond so that the client does not have to wait for a response that a network connection is open. The agent may open the network connection when required or store a cache of open network connections to the server. Also, the client hook may intercept requests from the client to compress information into changes from information in a previous request. The agent has the previous information stored and reconstructs the hew information from the changes. Thus, the communication between the client hook and the agent increases performance of communication between the client and the server.

A feature of the present invention is that performance is increased without necessitating modification of the client or server. As no modifications of a Web browser is required, the enhancements may be implemented to transparently increase the performance of the browser, regardless of the browser that is utilized. Other features and advantages of the present invention will become apparent upon a perusal of the remaining portions of the specification and drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates an example of a computer system used to execute software of an embodiment of the present invention;

FIG. **2** shows a system block diagram of a typical computer system used to execute software of an embodiment of the present invention;

FIG. **3** shows a diagram of multiple computers networked over the Internet;

FIG. **4** is a block diagram of a browser connected to a Web server through network access equipment including an agent;

FIG. **5** shows a high level flowchart of a process of opening multiple network connections to the server in response to a request to open a single network connection;

FIG. **6** shows a flowchart of another process of opening multiple network connections to the server in response to a request to open a single network connection utilizing a cache;

FIG. **7A** and **7B** show flowcharts of a process of increasing performance of requests to get an object on the server if it has been modified utilizing a cache;

FIG. **8** shows a flowchart of a process of periodically refreshing information in the cache utilized in FIGS. **7A** and **7B**;

FIG. **9** is a block diagram of a browser connected to a Web server through network access equipment which utilizes a proxy server to increase performance;

FIG. **10** shows a flowchart of a process of directing an HTTP request to either the Web server or the proxy server depending on the request;

FIG. **11** is a block diagram of a browser connected to a Web server through network access equipment in which a client hook intercepts requests from the browser;

FIG. **12** shows a flowchart of a process of a client hook immediately responding that network connection has been opened in response to a request to open a network connection to the server;

FIG. **13** shows a flowchart of a process of an agent receiving a request from the client hook that includes a request from the client and an identifier for the server to which the request should be sent;

FIG. **14** shows a flowchart of a process of an agent storing a header and reconstructing another header from the differences between the headers; and

FIG. **15** shows a flowchart of a process of a client hook and agent increasing the performance of header transmission.

DESCRIPTION OF PREFERRED EMBODIMENTS

In the description that follows, the present invention will be described in reference to preferred embodiments that increase the performance of Web browsers utilizing a weak link to network access equipment. The present invention, however, is not limited to any particular embodiment or computer network. Therefore, the description the embodiments that follow is for purposes of illustration and not limitation.

FIG. **1** illustrates an example of a computer system used to execute software of an embodiment of the present invention. FIG. **1** shows a computer system **1** which includes a monitor **3**, screen **5**, cabinet **7**, keyboard **9**, and mouse **11**. Mouse **11** may have one or more buttons such as mouse buttons **13**. Cabinet **7** houses a CD-ROM drive **15**, a system memory and a hard drive (see FIG. **2**) which may be utilized to store and retrieve software programs incorporating computer code that implements the present invention, data for use with the present invention, and the like. Although a CD-ROM **17** is shown as an exemplary computer readable storage medium, other computer readable storage media including floppy disks, tape, flash memory, system memory, and hard drives may be utilized. Cabinet **7** also houses familiar computer components (not shown) such as a central processor, system memory, hard disk, and the like.

FIG. **2** shows a system block diagram of computer system **1** used to execute the software of an embodiment of the present invention. As in FIG. **1**, computer system **1** includes monitor **3** and keyboard **9**. Computer system **1** further includes subsystems such as a central processor **102**, system memory **104**, I/O controller **106**, display adapter **108**, removable disk **112** (e.g., CD-ROM drive), fixed disk **116** (e.g., hard drive), network interface **118**, and speaker **120**. Other computer systems suitable for use with the present invention may include additional or fewer subsystems. For example, another computer system could include more than one processor **102** (i.e., a multi-processor system) or a cache memory.

Arrows such as **122** represent the system bus architecture of computer system 1. However, these arrows are illustrative of any interconnection scheme serving to link the subsystems. For example, a local bus could be utilized to connect the central processor to the system memory and display adapter. Computer system **1** shown in FIG. **2** is but an example of a computer system suitable for use with the present invention. Other configurations of subsystems suitable for use with the present invention will be readily apparent to one of ordinary skill in the art.

Preferred embodiments of the invention increase the performance of Web browsers' (or clients') access to the Web on the Internet. FIG. **3** shows a diagram of multiple computers networked over the Internet. Computers **202, 204** and **206** are interconnected by the Internet **208**, which is a series of high-speed communications links between educational, research and commercial computer sites around the world. Internet computers use the TCP/IP as the communications protocol.

The Web utilizes the HTTP client/server protocol, which is a request-response protocol. HTTP transactions include four stages: connection, request, response, and disconnection. In the connection stage, the client attempts to open a network connection to the server. Unless otherwise specified, HTTP attempts to use port **80** on the server for this connection. Establishing a connection involves one round-trip time from the client to the server as the client requests to open a network connection and the server responds that a network connection has been opened. Although the discussion herein focuses on version 1.0 of HTTP, the invention is not limited to any version of HTTP or to HTTP specifically.

After a network connection is open, the client may send an HTTP request to the server in the request stage. A request stage involves one half of a round-trip time as the request goes from the client to the server. Once the server receives the request, the server responds by sending a response to the client in the response stage. As with the request, the response stage involves one half of a round-trip time as the response goes from the server to the client.

The disconnection stage closes the network connection to the server. This stage involves one half of a round-trip time and may occur many different ways. The server may close the connection after the response is sent or by the client by sending a Stop sequence (e.g., the user clicked on the Stop button in the browser or the Back/Forward buttons). Conventional browsers show each of the four stages on a status line on the screen.

The terms of "client" and "server" are relative terms. A client is an entity that is making a request to a server which typically responds back to the client. However, these labels are request-response specific and are not an indication that the entities' roles are fixed. In preferred embodiments, the client is a browser and the server is a Web server. The browser may be executed on a computer similar to the one shown in FIGS. **1** and **2**. The server may be similar but is typically a much more powerful system including faster subsystems and more storage capacity.

FIG. **4** is a block diagram of a browser connected to a Web server through network access equipment including an agent. The computer network shown includes a Web browser **252**, network access equipment **254** and a Web server **256**. The browser communicates over a link to the network access equipment via a network protocol (e.g., TCP/IP) stack **258**. The browser and network protocol stack reside on the client computer system. The network access equipment is typically an electronic box and may include some of the subsystems shown in FIG. **2**. The Web server resides on a server which is typically a remote computer system.

The network access equipment includes an agent **260**. The agent is a program that includes embodiments of the invention. The computer code for the agent may reside on any computer readable storage medium including dynamic random access memory, electrically erasable programmable read only memory, or flash memory just to name a few. In a preferred embodiment, the agent resides on a LanRover™ Access Switch remote access server available from Shiva Corporation, Bedford, Mass.

FIG. **5** shows a high level flowchart of a process of opening multiple network connections to the server in response to a request to open a single network connection. The process shown is executed by an agent on the network access equipment. At step **302**, the agent receives a client request to open a single network connection to the server.

In response to the client request to open a single network connection to the server, the agent sends multiple requests to the server to open multiple network connections to the server at step **304**. Thus, multiple network connections to the server are opened in response to a client request to open a single network connection. Preferably, the agent requests persistent network connections. Once one of the network connections is open, the agent will receive a response from the server and send that response to the client. The client will then issue a request to the server over the open network connection which will be received by the agent. At step **306**, the agent sends the following client request to the server using the open network connection.

Oftentimes, the agent will receive another client request to open a single network connection to the server. Since the agent previously opened multiple network connections, the agent responds immediately that a network connection is available, thus saving a round-trip time between the agent and server. The client then issues the following client request over the open network connection. The agent may store the open network connections in a cache, which will be described more detail in reference to FIG. **6**.

For simplicity, the discussion herein describes the interaction of the agent with a single client browser and a single Web server. However, in practice, the agent is typically in communication with multiple clients and multiple Web servers. The methods of the present invention are not isolated to increasing the performance of each individual client alone. For example, one client may open multiple network connections to the server by issuing a request to open a single network connection. Subsequently, another client may request to open a single network connection to the same server. The agent may then immediately grant a network connection to this client as a network connection has already been opened. Thus, the actions of one client may also result in an increase in performance of other clients. The agent preferably opens another network connection to the server to replace the one that has become used.

FIG. **6** shows a flowchart of another process of opening multiple network connections to the server in response to a request to open a single network connection utilizing a cache. In this embodiment, the agent maintains a cache of network connections to the server (or servers). At step **352**, the agent receives a client request to open a single network connection to the server.

The agent scans the cache for an open network connection to the server at step **354**. If an open network connection to the server is not available in the cache at step **356**, the agent sends a request to the server to open a network connection. Although this embodiment opens a single network connection at this point and makes a subsequent determination if connection caching improves performance, in another embodiment, the agent sends multiple requests to the server to open multiple network connections and bypasses the subsequent determination.

If an open network connection to the server is available in the cache, the agent sends a response to the client that a network connection is open. The client sends a client request to the server using the open network connection which the agent sends to the server at step **360**.

At step **362**, the agent determines if network connection caching for the server improves performance. This determination may be made from many factors including the number of times the agent has a "hit" in the cache, the overhead required to maintain the cache, whether the server is allowing the network connections to stay open in response to a request to "keep open" the connection (i.e., persistent connection), and the like. The agent may request that the server "keep open" the connection but honoring this request is at the discretion of the server.

The agent checks if all the network connections to the server in the cache are in use at step **364**. If connection caching does improve performance and all the network connections to the server are in use, the agent sends at least one request to the server to open a network connection at step **366**. Multiple network connections may be opened to the server if it has been determined that this improves performance. For example, it may be beneficial to have a predetermined number (e.g., user specified or determined by the agent as it monitors performance) of network connections open in the cache for a server. If the number of open connections is less than the predetermined number, the agent sends at least one request to open a network connection to the server.

In a preferred embodiment, the cache stores all the network connections and an indication of whether the network connection is open, in use or closed. As the agent opens network connections, they are marked as "open." When the agent receives a request to open a network connection to the server and there is an open network connection to the server in the cache, the agent marks the network connection as "in use" or "used." When the agent receives an indication from the server that a network connection in the cache has been closed, the agent marks the network connection as "closed."

In another embodiment, the cache stores only open network connections. Each time a network connection in the cache is either used or closed, the agent removes the network connection from cache. The agent may also issue a request to the server to open a network connection to replace a network connection removed from the cache.

In conventional network systems, when a client wants to get an object from the server, it takes two round-trip times between the client and server: one to open the connection and one to get the object. With the present invention, one round-trip time between the agent and server may be avoided, thus reducing the overall time to one and a half round-trip times between the client and server. This provides a significant increase in the performance of the client.

FIG. **7**A and **7**B show flowcharts of a process of increasing performance of requests to get an object on the server if it has been modified utilizing a cache. With HTTP, a client is able to request that the server send an object if it has not been modified since a specified time and date, which will be herein called a timestamp for convenience. More specifically, the header of an HTTP get message may include a header with fields specifying "If-Modified-Since" in one field and a timestamp in another field. If the server determines that the object has not been modified since the specified timestamp, the server does not need to send the object to the client but instead issues a not-modified **(304)** response.

Although this feature may be utilized by the browser to maintain its own cache, an agent of the present invention utilizes the feature to maintain a cache of information to further increase the performance of the computer network.

In general, it may take two client requests to set up the cache of information and a third client request to realize a performance increase. For completeness, the following assumes that browser does not have the desired object in its cache.

At step **402** in FIG. **7**A, the agent receives a client request to get an object from the server. The agent then sends the client request to the server. The agent receives the object from the server and sends it to the client at step **404**. Conventional browsers have a cache in which they store objects for future reference (e.g., when a user revisits the web page). The browser cache includes timestamps indicating the currency of the objects in the cache.

When the client desires an object in its cache, the browser sends a request to get the object if it has been modified since the timestamp specified in the browser cache. The agent receives this request and sends it to the server at step **406**.

At step **408**, the agent receives a response **(304)** from the server that the object has not been modified since the timestamp. The agent sends this response to the client. The agent stores an identifier for the object and the current timestamp in a cache (i.e., the timestamp of when the server indicated that the object had not changed) at step **410**. The current timestamp will be utilized as an estimate of the time at which the object remained unmodified. The cache may be a table including the address of the object (e.g., an identifier), the timestamp of the object in the browser's cache and the current timestamp. The agent does not need to store the object in the cache.

The agent receives a request to get the object from the server if it has been modified at step **412**. The agent determines if the request specifies an object in its caches by scanning the cache. As the previous client request described above requested this same object and received a not-modified response, the object is specified in the cache.

Now referring to FIG. **7**B, the agent determines if policy indicates the object in the browser's cache is sufficiently current at step **414**. The policy may be a comparison of the current timestamp to the timestamp in the cache of when the server last indicated that the object had not been modified. If the difference between these timestamps is within a predetermined amount of time, the object in the browser's cache is sufficiently current. The predetermined time may be set by a administrator or may be preset by the agent. Additional policy considerations may be applied. If the server does not change its contents often (e.g., as noticed by the agent), the amount of time may be lengthened. On the other hand, if the server does change its contents often (e.g., stock quotes), the amount of time may be shortened. Thus, the amount of time for an object still being current may be server, Web page or Uniform Resource Locator ("URL") specific.

At step **416**, the agent has determined that the object in the browser's cache is sufficiently current and the agent sends a not-modified response to the client. The agent responds to the client without sending a request to the server, thereby saving a round-trip time between the agent and server.

If the agent determines that the browser's cache is not sufficiently current, the agent sends a request to get the object form the server if it has been modified at step **418**. Thus, the agent sends the client request to the server. When the agent receives a response from the server, the agent sends the response to the client at step **420**. The agent updates the cache according to the response at step **422**. For example, the agent may store the current timestamp in the cache to indicate that at this point in time the server indicated that the object had not been modified. If a new copy of the object is received, the agent may also update the timestamp in the cache indicating the last time the browser received the object.

With the invention, the time for a client requesting an object if it has been modified may be reduced from one round-trip time between the client to the server to a round-trip time between the client and the agent (ignoring connection and disconnection times for the moment). Although the entries in the cache are client specific, the invention provides a significant performance increase for the clients when they issue a request to get an object if it has been modified.

FIG. **8** shows a flowchart of a process of periodically refreshing information in the cache utilized in FIGS. **7**A and **7**B. Periodically (e.g., using a timer), the agent gets an identifier for an object in the cache at step **452**. The agent then makes a determination of whether the object is sufficiently current at step **454**. This may be done by the agent making the same calculation as if a client requested to get the object if it has been modified.

If the object is not sufficiently current, the agent sends a request to the server to get the object if it has been modified at step **456**. This request originates from the agent and not the client. The agent then updates its cache at step **458** depending on the response from the server. If the server responds that the object has not been modified, the agent may update the estimate of the time at which the object remained unmodified. Otherwise, if the server sends a new copy of the object, the agent typically discards the new object and updates the cache to indicate the object has been modified. In other embodiments, the agent may store the new copy of the object in order to fulfill future client requests.

At step **460**, the agent determines if there is another object identified in the cache. If there is, the agent tries to update the cache for that object. By periodically updating the cache, the performance of the client browser will increase as more round-trip times between the agent and server may be eliminated.

FIG. **9** is a block diagram of a browser connected to a Web server through network access equipment which utilizes a proxy server to increase performance. The computer network shown includes a Web browser **502**, network access equipment **504** and a Web server **506**. The browser communicates over a link to the network access equipment via a network protocol stack **508**. The browser and network protocol stack reside on the client computer system. The Web server resides on a server which is typically a remote computer system.

The network access equipment includes an agent **510**. The agent shown is a program that receives HTTP requests and directs them to either the Web server or a proxy server **512**. The agent will typically receives messages in a number of protocols but the discussion herein will focus on HTTP messages. The proxy server is a computer system that stores information available from the Web server. In general, it may be quicker to access information from the proxy server instead of the Web server.

Although the use of proxy servers is known, conventional systems require the client to specify whether an HTTP message be sent to the Web server or the proxy server. With the present invention the client need not explicitly specify the proxy server to gain an increase in performance resulting from use of the proxy server. The agent of the present invention sends the HTTP requests to either the Web server or proxy server based on an analysis of the HTTP request.

FIG. **10** shows a flowchart of a process of directing an HTTP request to either the Web server or the proxy server depending on the request. At step **552**, the agent receives an HTTP request from the client. The agent analyzes the HTTP request at step **554**. The analysis may include a determination of whether the request gets information or posts information. Requests that post information may be sent to the Web server. However, requests that get information may be sent to the proxy server. There may also be other factors including determining if this information is likely to reside on the proxy server.

If it is determined that the HTTP request may be serviced by the proxy server at step **556**, the agent sends the HTTP request to the proxy server at step **558**. The agent may also need to translate the request to a different protocol before it is sent to the proxy server. Otherwise, the agent sends the HTTP request to the Web server at step **560**.

The invention allows a client to obtain the benefits of a proxy server without needing to be modified to send requests explicitly to the proxy server. Thus, the proxy server may be changed or otherwise modified by effecting changes to the network access equipment while the client remains unchanged.

FIG. **11** is a block diagram of a browser connected to a Web server through network access equipment in which a client hook intercepts requests from the browser. The computer network shown includes a Web browser **602**, network access equipment **604** and a Web server **606**. The browser communicates over a link to the network access equipment via a network protocol stack **608**. On the client computer system with the browser and network protocol stack is a client hook **610**, the client hook intercepts calls between the browser and the network protocol stack.

In preferred embodiment, the client hook intercepts calls between the browser and the network protocol stack utilizing DLL chaining. For example, the dynamic link library WINSOCK.DLL is renamed to WINSOCKZ.DLL. A new WINSOCK.DLL is installed on the client computer system that has routines with the same name as in the original WINSOCK.DLL. However, the new WINSOCK.DLL has instructions in the routines (i.e., the client hook) to intercept calls before they are executed. In many instances, the routines in WINSOCK.DLL call the routines in WINSOCKZ.DLL at some point in the routine.

The network access equipment includes an agent **612**. The agent shown is a program that receives HTTP requests from the client hook. The client hook and agent communicate in such a way to increase performance of the computer network without requiring a modification of the client. Accordingly, a user is free to select the browser of his or her choice and still receive a significant performance increase.

In general, the client hook intercepts HTTP requests from the client to the server. The client hook modifies the HTTP requests from the client and sends the modified requests to the agent. The agent receives the modified requests and reconstructs the original HTTP requests from the client according to the modified requests. The agent then sends the HTTP requests from the client to the server. There is no requirement that the client hook and agent communicate via HTTP. Nevertheless, it is the communication between the client hook and the agent increases performance of communication between the client and the server.

The computer network shown in FIG. **11** may be utilized to increase performance of many procedures. For example, the procedure of opening a network connection to the server may be improved. Additionally, the procedure of sending headers within requests to the server may enhanced. These are but a couple examples of the present invention which will be described in more detail in reference to FIGS. **12–15**.

FIG. 12 shows a flowchart of a process of a client hook immediately responding that network connection has been opened in response to a request to open a network connection to the server. At step **622**, the client hook intercepts a client request to open a network connection to the server. The client hook immediately responds to the client that a dummy network connection has been opened at step **624**. The dummy network connection is not an actual network connection but allows the client to proceed with the next client request.

The client hook intercepts a client request to the server that specifies the dummy network connection at step **626**. At step **628**, the client hook sends the client request and an identifier for the server to the agent. The identifier for the server (e.g., the address) is obtained from the client request to open a network connection. As the client hook and agent are in communication within the computer network, there is no requirement that the messages between the two conform to HTTP. Thus, the actual protocol utilized may be optimized for the actual link.

FIG. **13** shows a flowchart of a process of an agent receiving a request from the client hook that includes a request from the client and an identifier for the server to which the request should be sent. At step **652**, the agent receives a request from the client hook which includes an identifier for the server. The agent receives the request from the client hook without necessarily first receiving an HTTP client request to open a network connection to the server.

The agent generates and sends an HTTP request to the server to open a network connection at step **654**. Preferably, the agent requests a persistent network connection. The server is identified by the identifier received from the client hook. Once the agent receives a response from the server that a network connection is open, the agent generates and sends the client request in the form of an HTTP request to the server at step **656**.

The agent may send an HTTP request to open a network connection to the server. The agent may also maintain a cache of network connections as was described in reference to FIG. **6**. In this manner, a round-trip time between the agent and server may be eliminated.

The invention increases performance in many ways. A round-trip time between the client and agent may be eliminated when opening a network connection to the server. This may be especially significant because this link may be the weak link in the computer network. Additionally, the protocol between the client hook and agent is not restricted to HTTP so it may more optimized.

FIG. **14** shows a flowchart of a process of an agent storing a header and reconstructing another header from the differences between the headers. Initially, the client hook intercepts client requests and sends them to the agent. As will be described below, the communication between the client hook and agent is preferably not HTTP as it is optimized. Additionally, although the embodiment described is directed to headers in the requests, the invention is applicable to any information within the requests.

When the agent receives a request from the client hook, the agent stores the header at step **672**. The agent then generates and sends a corresponding HTTP request to the server. At step **674**, the agent receives another request from the client hook that includes differences between the previous header and this header. The differences between headers is not currently a standard format of headers in HTTP.

The agent reconstructs the header for the current request from the client hook utilizing the stored header and the differences at step **676**. At step **678**, the agent uses the reconstructed header in generating and sending a corresponding HTTP request to the server.

The header typically includes information about the browser (e.g., name and version number), acceptable data formats, and the like similar to a Multipurpose Internet Mail Extensions ("MIME") header. Accordingly, much of the header does not change from request to request. With the invention, an HTTP request may be reduced from several hundred bytes to a request that is less than twenty bytes. This is especially significant as the link between the client hook and agent is typically the weak link in the computer network. The following describes an embodiment of this process in more detail.

FIG. **15** shows a flowchart of a process of a client hook and agent increasing the performance of header transmission. At step **702**, the client hook intercepts a client request to the server that includes a header, which the client hook stores. The client request is an HTTP request and the client hook utilize a more optimized protocol (i.e., non-HTTP) when it sends the client request to the agent.

The client hook sends the client request to the agent for transmission to the server at step **704**. Once the agent receives the client request, the agent stores a copy of the header in the client request at step **706**. If the client request is non-HTTP, the agent generates a corresponding HTTP client request. The agent sends the client request to the server at step **708**.

At step **710**, the client hook intercepts a client request to the server that includes a header. The client hook modifies the client request to include a header that specifies differences between this header and the previous header at step **712**. Thus, the client request will contain the differences or deltas between the headers.

The agent receives the modified client request at step **714**. With the modified client request, the agent reconstructs the header from the stored header and the differences between the headers at step **716**. The agent generates an HTTP request that corresponds to the client request and includes the reconstructed header. The agent sends the client request to the server at step **718**.

The invention increases performance in many ways. A round-trip time between the client and agent may be eliminated when opening a network connection to the server. This may be especially significant because the link between the client hook and agent may be substantially slower than the link between the agent and server. Additionally, the protocol between the client hook and agent is not restricted to HTTP so it may be more optimized.

While the above is a complete description of preferred embodiments of the invention, various alternatives, modifications and equivalents may be used. It should be evident that the present invention is equally applicable by making appropriate modifications to the embodiments described above. For example, although the embodiments have been described individually, many of the embodiments may be combined to further increase performance. Therefore, the above description should not be taken as limiting the scope of the invention which is defined by the metes and bounds of the appended claims along with their full scope of equivalents.

What is claimed is:

1. In a computer network, a method executed by an agent in the computer network between clients and a server for increasing performance between the clients and the server, the method comprising the steps of:

receiving a first request from a client to open a single network connection to the server;

sending a plurality of requests to the server to open a plurality of network connections to the server;

receiving a second request from the client;

sending the second request to the server using one of the plurality of network connections;

wherein the plurality of network connections to the server are opened in response to the first request from the client to open a single network connection.

2. The method of claim **1**, further comprising the steps of:

receiving a third request from a client to open a single network connection to the server;

sending a response to the client that a network connection is open;

receiving a fourth request from the client; and

sending the fourth request to the server using one of the plurality of network connections previously obtained in response to the first request.

3. The method of claim **2**, further comprising the step of sending a request to the server to open a network connection to the server in order to replace the one of the plurality of network connections being used.

4. The method of claim **1**, further comprising the step of storing the plurality of network connections in a cache of network connections.

5. The method of claim **4**, further comprising the steps of:

receiving a third request from a client to open a single network connection to the server;

scanning the cache to determine if there is an open network connection to the server in the cache; and

if there is an open network connection in the cache, sending a response to the client that a network connection is open, whereby the open network connection becomes used.

6. The method of claim **5**, further comprising the steps of:

determining if network connection caching increases performance between the clients and the server; and

if network caching increases performance, sending a request to the server to open a network connection to the server in order to store another open network connection in the cache to replace the used network connection.

7. The method of claim **4**, further comprising the steps of:

determining the number of open network connections to the server stored in the cache; and

if the number of open network connections is less than a predetermined number, sending a request to the server to open a network connection to the server in order to store another open network connection in the cache.

8. The method of claim **4**, further comprising the steps of:

determining if an open network connection to the server in the cache has been closed; and

if there is a closed network connection in the cache, sending a request to the server to open a network connection to the server in order to store an open network connection in the cache.

9. The method of claim **5**, further comprising the step of removing the used network connection from the cache.

10. The method of claim **5**, further comprising the step of sending a request to the server to open a network connection to the server in order to store another open network connection in the cache to replace the used network connection.

11. The method of claim **1**, wherein the client is a World Wide Web browser.

12. A computer network, comprising:

a client computer running a Web browser;

a Web server networked to the client computer;

a proxy server computer networked to the client computer for storing information available on the Web server; and

network access equipment, networked between the client computer and the Web and proxy servers, including an agent that receives an HTTP request from the Web browser to open a single network connection to the server and sends a plurality of requests to the server to open a plurality of network connections to the server;

wherein the plurality of network connections to the server are opened in response to the HTTP request from the Web browser to open a single network connection.

13. In a computer network, a method executed by an agent in the computer network between clients and a server for increasing performance between the clients and the server, the method comprising the steps of:

receiving a first request from a client to get an object from the server if the object has been modified after a specific timestamp;

sending the first request to the server;

receiving a first response from the server that the object has not been modified after the specific timestamp;

sending the first response to the client;

storing an identifier for the object and a timestamp in a cache;

receiving a second request from the client to get the object from the server if the object has been modified after the specific timestamp; and

if the timestamp stored in the cache is within a predetermined amount of time from the current time, sending a second response to the client that the object has not been modified after the specific timestamp without sending the second request to the server.

14. The method of claim **13**, wherein the storing step includes the steps of storing a location of the object as the identifier, storing the specific timestamp, and storing the timestamp as the current time in order to estimate at what time the object remained unmodified.

15. The method of claim **14**, further comprising the step of periodically sending requests to the server to get objects identified in the cache if the object has been modified after the specific timestamp in order to update the timestamp in the cache.

16. The method of claim **13**, further comprising the step of setting the predetermined amount of time.

17. The method of claim **13**, wherein the client is a World Wide Web browser.

18. A computer network, comprising:

a client computer running a Web browser;

a Web server networked to the client computer;

a proxy server computer networked to the client computer for storing information available on the Web server; and

network access equipment, networked between the client computer and the Web and proxy servers, including an agent that stores identifiers and timestamps for objects so that when the agent receives a request from the Web browser to get an object from the server if the object has been modified after a specific timestamp, the agent responds to the request without sending a request to the Web server.

19. In a computer network, a method executed by an agent in the computer network between a client and a Web and proxy servers for increasing performance between the client and the Web server, comprising the steps of:

receiving an HTTP request from a client; and

sending the HTTP request to either the Web server or the proxy server depending on the HTTP request, the proxy server storing information available on the Web server;

wherein the client does not need to be modified to utilize the proxy server.

20. The method of claim 19, wherein if the HTTP request may be serviced by the proxy server, the HTTP request is sent to the proxy server, and otherwise the HTTP request is sent to the Web server.

21. The method of claim 19, wherein if the HTTP request is to post information to the server, the HTTP request is sent to the Web server, and otherwise the HTTP request is sent to the proxy server.

22. The method of claim 19, further comprising the step of translating the HTTP request to a different protocol before the HTTP request is sent to the proxy server.

23. The method of claim 19, wherein the client is a World Wide Web browser.

24. A computer network, comprising:

a client computer running a Web browser;

a Web server networked to the client computer;

a proxy server computer networked to the client computer for storing information available on the Web server; and

network access equipment, networked between the client computer and the Web and proxy servers, including an agent that receives HTTP requests and sends the HTTP requests to either the Web server or the proxy server depending on each HTTP request;

wherein software on the client computer does not need to be modified to utilize the proxy server.

25. The computer network of claim 24, wherein HTTP requests that may be serviced by the proxy server are sent to the proxy server, otherwise the HTTP requests are sent to the Web server.

26. The computer network of claim 24, wherein HTTP requests that post information to the server are sent to the Web server, otherwise the HTTP requests are sent to the proxy server.

27. The computer network of claim 24, wherein the HTTP requests are translated to a different protocol before the HTTP requests are sent to the proxy server.

28. In a computer network, a method for increasing performance between a client on a client computer and a server utilizing a client hook on the client computer and an agent between the client computer and the server, comprising the steps of:

the client hook intercepting requests from the client to the server;

the client hook modifying the requests from the client;

the client hook sending the modified requests to the agent;

the agent reconstructing the requests from the client according to the modified requests; and

the agent sending the requests from the client to the server;

wherein communication between the client hook and the agent increases performance of communication between the client and the server.

29. The method of claim 28, further comprising the steps of:

the client hook intercepting a first request from the client to open a network connection to the server, the first request including an identifier for the server;

the client hook immediately responding that a network connection to the server has been opened to the server and storing the identifier of the server;

the client hook intercepting a second request from the client to be sent over the opened network connection to the server; and

the client hook sending the second request and the identifier of the server to the agent.

30. The method of claim 29, further comprising the steps of:

the agent sending a third request to open a network connection to the server identified by the identifier; and

the agent sending the second request to the server over an open network connection.

31. The method of claim 29, further comprising the steps of:

the agent identifying an open network connection to the server in a cache; and

the agent sending the second request to the server over the open network connection.

32. The method of claim 29, further comprising the steps of:

without first receiving a request from the client to open a network connection to the server, the agent receiving a first request from the client to the server and an identifier for the server;

utilizing the identifier for the server, the agent sending a second request to the server to open a network connection; and

the agent sending the first request to the server over an open network connection.

33. The method of claim 28, further comprising the steps of:

the agent storing first information included in a first request from the client to the server;

the agent receiving a second request from the client to the server that includes differences between the first information and second information of the second request instead of the second information;

the agent reconstructing the second information from the stored first information and the differences between the first and second information; and

the agent sending the second request to the server including the reconstructed second information.

34. The met hod of claim 28, further comprising the steps of:

the client hook intercepting a first request from the client to the server that includes first information;

the client hook sending the first request to the agent for sending to the server;

the agent storing a copy of the first information;

the agent sending the first request to the server;

the client hook intercepting a second request from the client to the server that includes second information;

the client hook modifying the second request to include differences between the first and second information instead of the second information;

the agent receiving the modified second request from the client;

the agent reconstructing the second information from the stored first information and the differences between the first and second information; and the agent sending the second request to the server including the reconstructed second information.

35. The method of claim 28, wherein the network link between the client computer and the agent is substantially slower than the network link between the agent and the server.

36. In a computer network, a method for increasing performance between a client on a client computer and a server utilizing a client hook on the client computer and an agent between the client computer and the server, comprising the steps of:

the client hook intercepting a first request from the client to open a network connection to the server, the first request including an identifier for the server;

the client hook immediately responding that a network connection to the server has been opened to the server and storing the identifier of the server;

the client hook intercepting a second request from the client to be sent over the opened network connection to the server; and

the client hook sending the second request and the identifier of the server to the agent.

37. The method of claim 36, further comprising the steps of:

the agent sending a third request to open a network connection to the server identified by the identifier; and

the agent sending the second request to the server over an open network connection.

38. The method of claim 36, further comprising the steps of:

the agent identifying an open network connection to the server in a cache; and

the agent sending the second request to the server over the open network connection.

39. The method of claim 36, wherein the second request to the server to open a network connection includes a request to keep the network connection open.

40. In a computer network, a method executed by an agent in the computer network between a client and a server for increasing performance between the client and the server, the method comprising the steps of:

without first receiving a request from the client to open a network connection to the server, receiving a first request from the client to the server and an identifier for the server;

utilizing the identifier for the server, sending a second request to the server to open a network connection; and

sending the first request to the server over an open network connection.

41. The method of claim 40, wherein the second request to the server to open a network connection includes a request to keep the network connection open.

42. The method of claim 40, wherein the client is a World Wide Web browser.

43. In a computer network, a method executed by an agent in the computer network between a client and a server for increasing performance between the client and the server, the method comprising the steps of:

storing first information included in a first request from the client to the server;

receiving a second request from the client to the server that includes differences between the first information and second information of the second request instead of the second information;

reconstructing the second information from the stored first information and the differences between the first and second information; and

sending the second request to the server including the reconstructed second information.

44. The method of claim 43, wherein the first and second requests are HTTP requests.

45. In a computer network, a method for increasing performance between a client on a client computer and a server utilizing a client hook on the client computer and an agent between the client computer and the server, comprising the steps of:

the client hook intercepting a first request from the client to the server that includes first information;

the client hook sending the first request to the agent for sending to the server;

the agent storing a copy of the first information;

the agent sending the first request to the server;

the client hook intercepting a second request from the client to the server that includes second information;

the client hook modifying the second request to include differences between the first and second information instead of the second information;

the agent receiving the modified second request from the client;

the agent reconstructing the second information from the stored first information and the differences between the first and second information; and

the agent sending the second request to the server including the reconstructed second information.

46. The method of claim 45, wherein the client hook intercepts requests from the client utilizing dynamic link library chaining.

47. The method of claim 45, wherein the first and second requests are HTTP requests.

48. The method of claim 45, wherein the client is a World Wide Web browser.

* * * * *

US006609154B1

(12) **United States Patent**
Fuh et al.

(10) **Patent No.:** **US 6,609,154 B1**
(45) **Date of Patent:** ***Aug. 19, 2003**

(54) LOCAL AUTHENTICATION OF A CLIENT AT A NETWORK DEVICE

(75) Inventors: **Tzong-Fen Fuh**, Fremont, CA (US); **Serene H. Fan**, Palo Alto, CA (US); **Diheng Qu**, Santa Clara, CA (US)

(73) Assignee: **Cisco Technology, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **10/264,655**

(22) Filed: **Oct. 3, 2002**

Related U.S. Application Data

(63) Continuation of application No. 09/347,433, filed on Jul. 2, 1999, now Pat. No. 6,463,474.

(51) **Int. Cl.**^{7} **G06F 15/173**

(52) **U.S. Cl.** **709/225**

(58) **Field of Search** 709/225, 229, 709/222, 223; 713/200, 201

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,991,807 A	*	11/1999	Schmidt et al.	709/225
6,182,142 B1	*	1/2001	Win et al.	709/229
6,219,706 B1	*	4/2001	Fan et al.	709/225
6,233,576 B1	*	5/2001	Lewis	707/9
6,233,618 B1	*	5/2001	Shannon	709/229
6,292,798 B1	*	9/2001	Dockter et al.	707/9
6,292,904 B1	*	9/2001	Broomhall et al.	714/1

* cited by examiner

Primary Examiner—Glenton B. Burgess
Assistant Examiner—Kimberly Flynn
(74) *Attorney, Agent, or Firm*—Hickman Palermo Truong & Becker LLP

(57) ABSTRACT

A method and apparatus that provide network access control are disclosed. In one embodiment, a network device is configured to intercept network traffic initiated from a client and directed toward a network resource, and to locally authenticate the client. Authentication is carried out by comparing information identifying the client to authentication information stored in the network device. In one embodiment, an authentication cache in the network device stores the authentication information. If the client identifying information is authenticated successfully against the stored authentication information, the network device is dynamically re-configured to allow network traffic initiated by the client to reach the network resource. If local authentication fails, new stored authentication is created for the client, and the network device attempts to authenticate the client using a remote authentication server. If remote authentication is successful, the local authentication information is updated so that subsequent requests can authenticate locally. As a result, a client may be authenticated locally at a router or similar device, reducing network traffic to the authentication server.

29 Claims, 9 Drawing Sheets

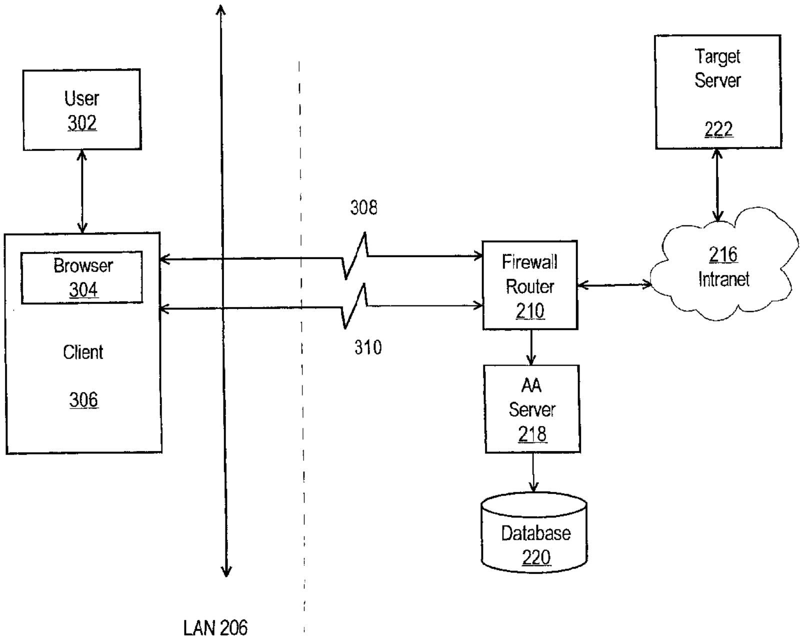
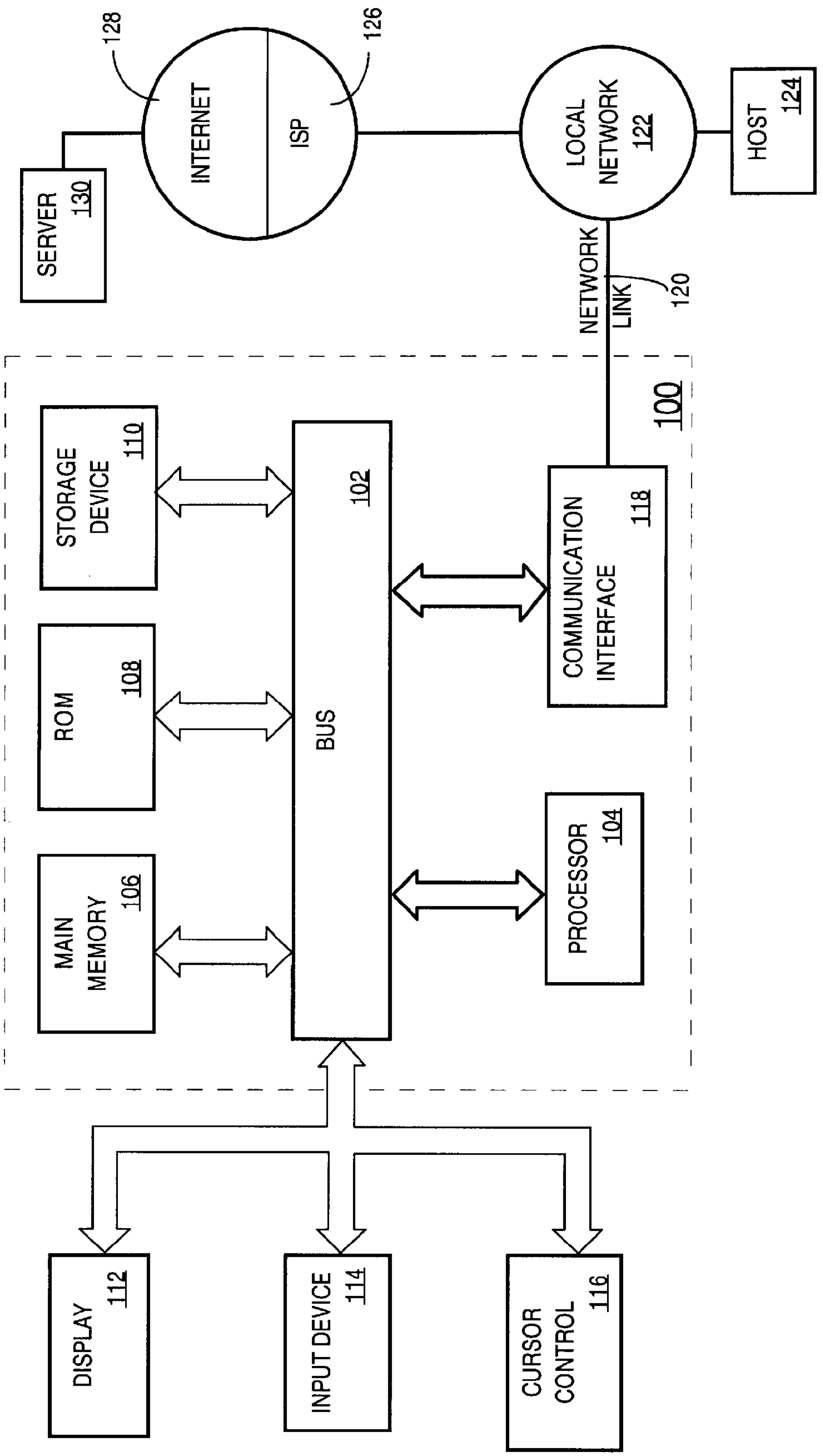


FIG. 1


SERVER 130
INTERNET
ISP
128
126
LOCAL NETWORK 122
HOST 124
NETWORK LINK
120
STORAGE DEVICE 110
ROM 108
MAIN MEMORY 106
BUS
102
COMMUNICATION INTERFACE 118
PROCESSOR 104
100
DISPLAY 112
INPUT DEVICE 114
CURSOR CONTROL 116

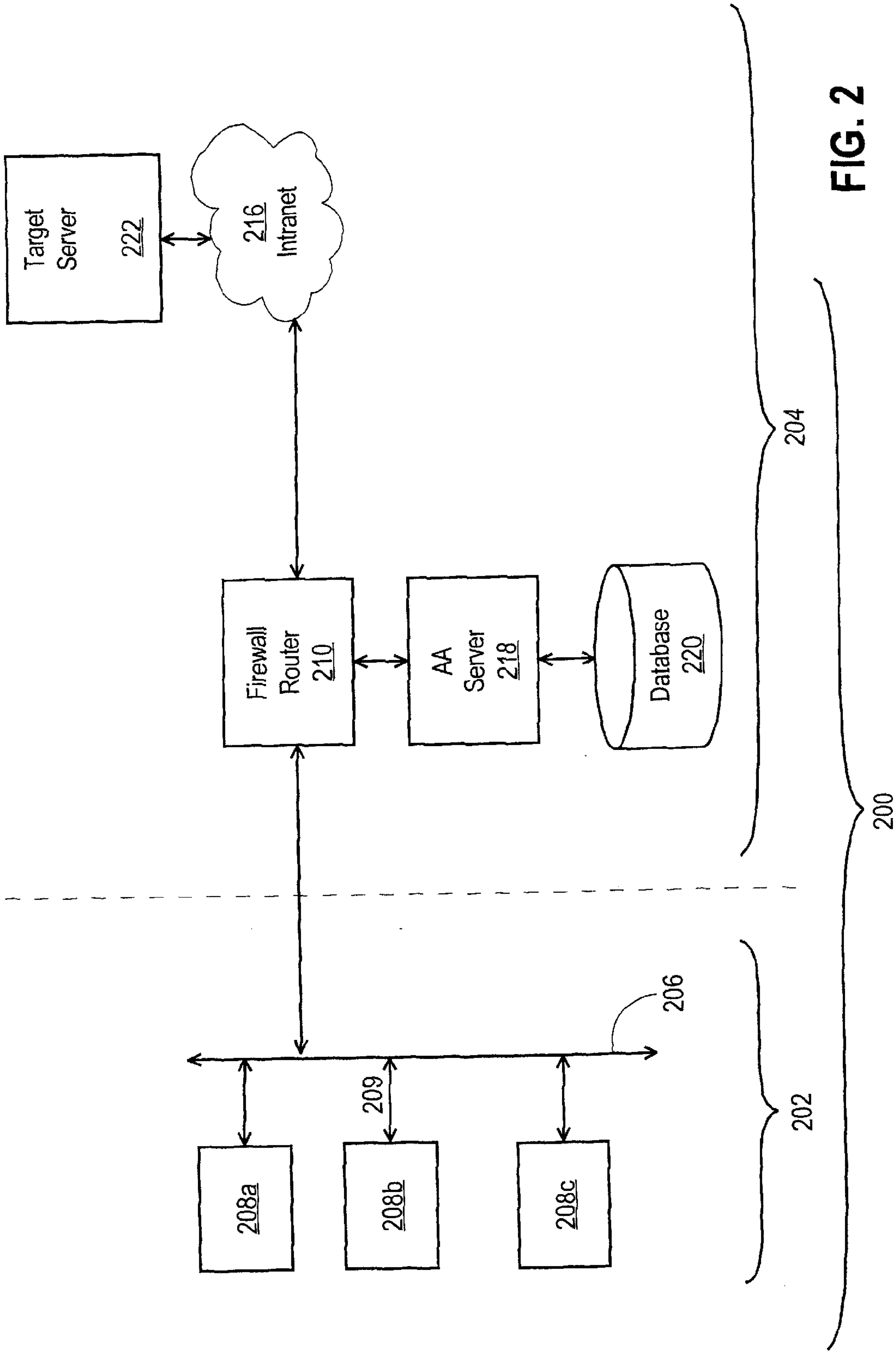

Target Server 222
216 Intranet
Firewall Router 210
AA Server 218
Database 220
204
200
206
209
208a
208b
208c
202

FIG. 2

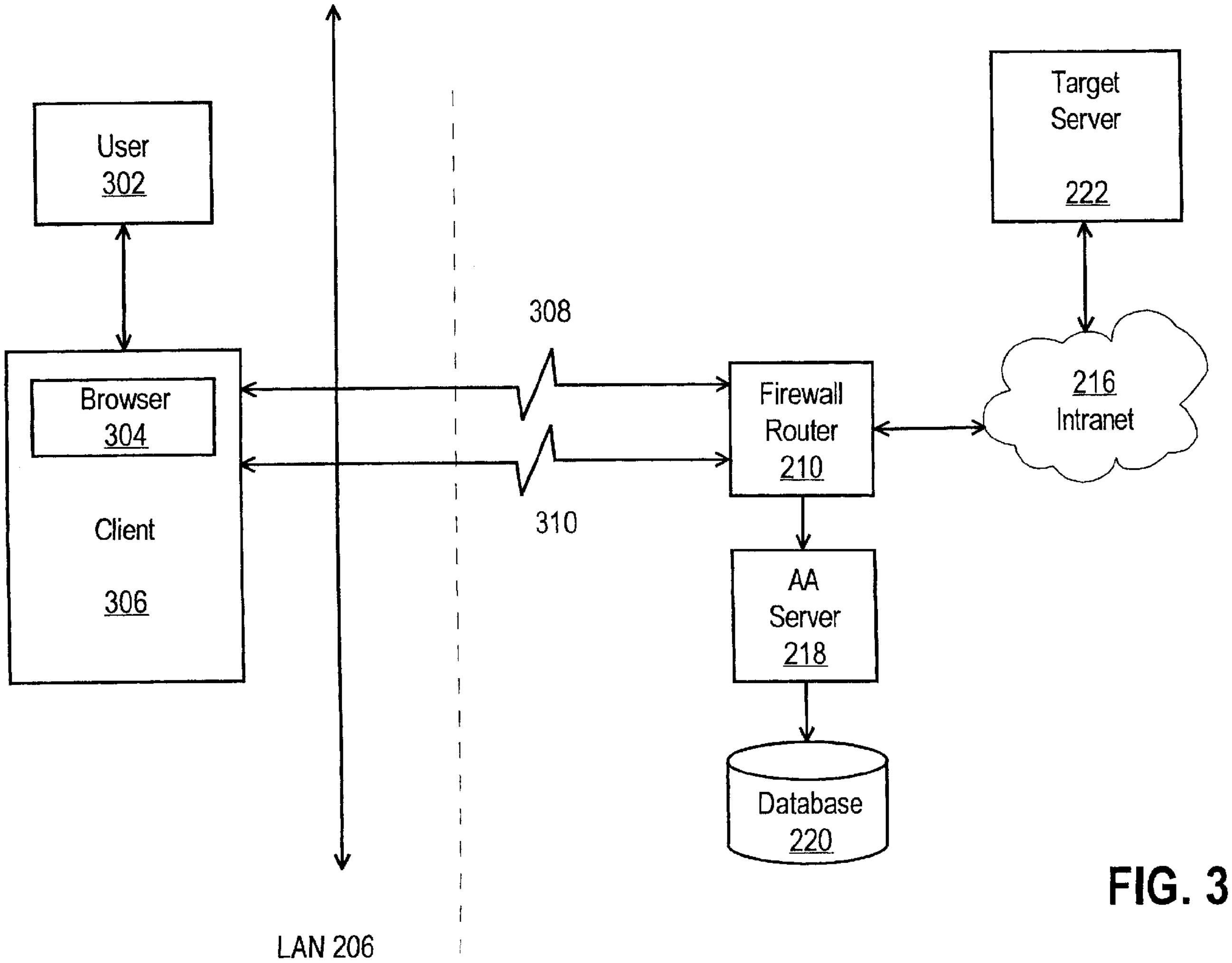

User
302
Browser
304
Client
306
LAN 206
308
310
Firewall
Router
210
AA
Server
218
Database
220
Target
Server
222
216
Intranet

FIG. 3

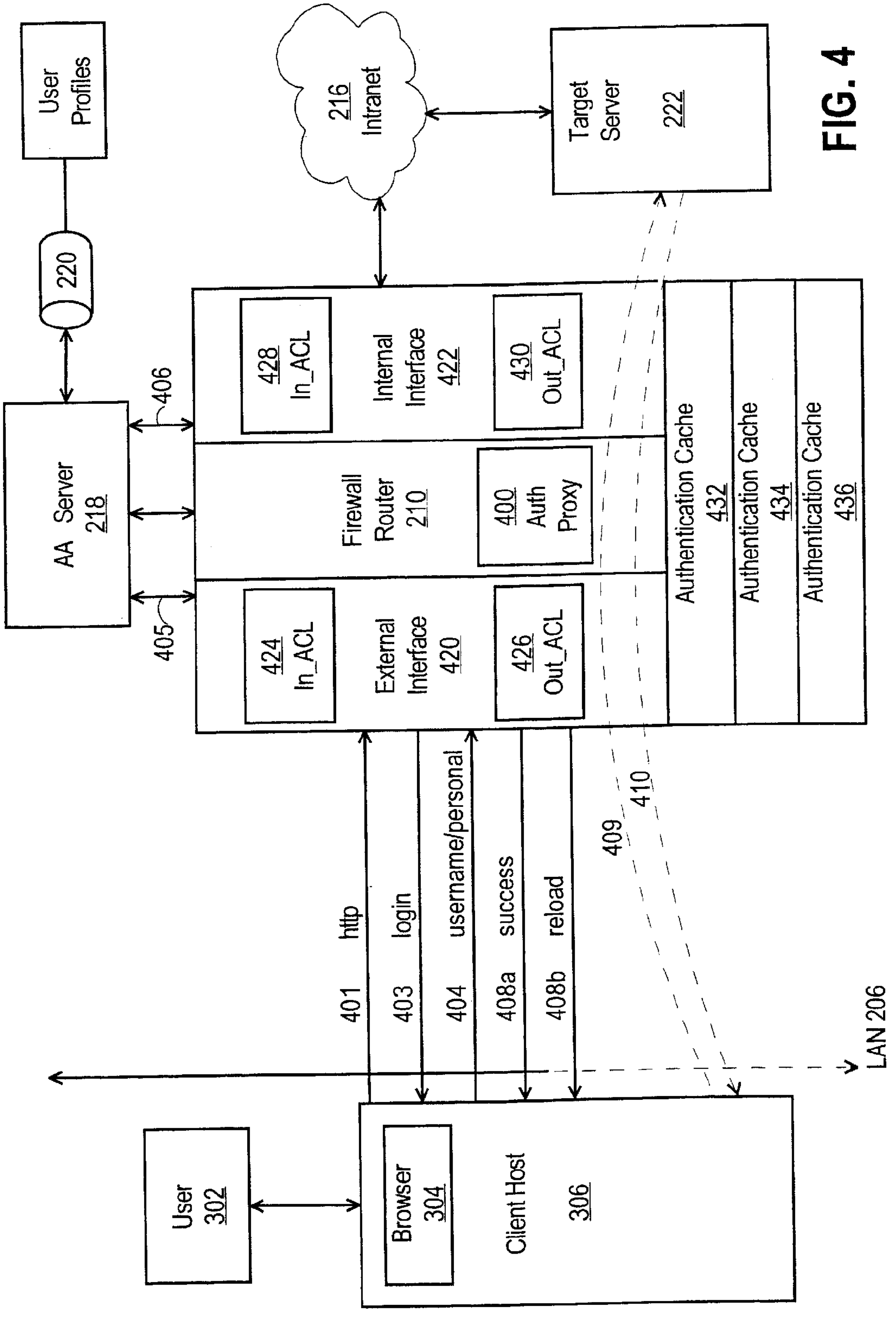
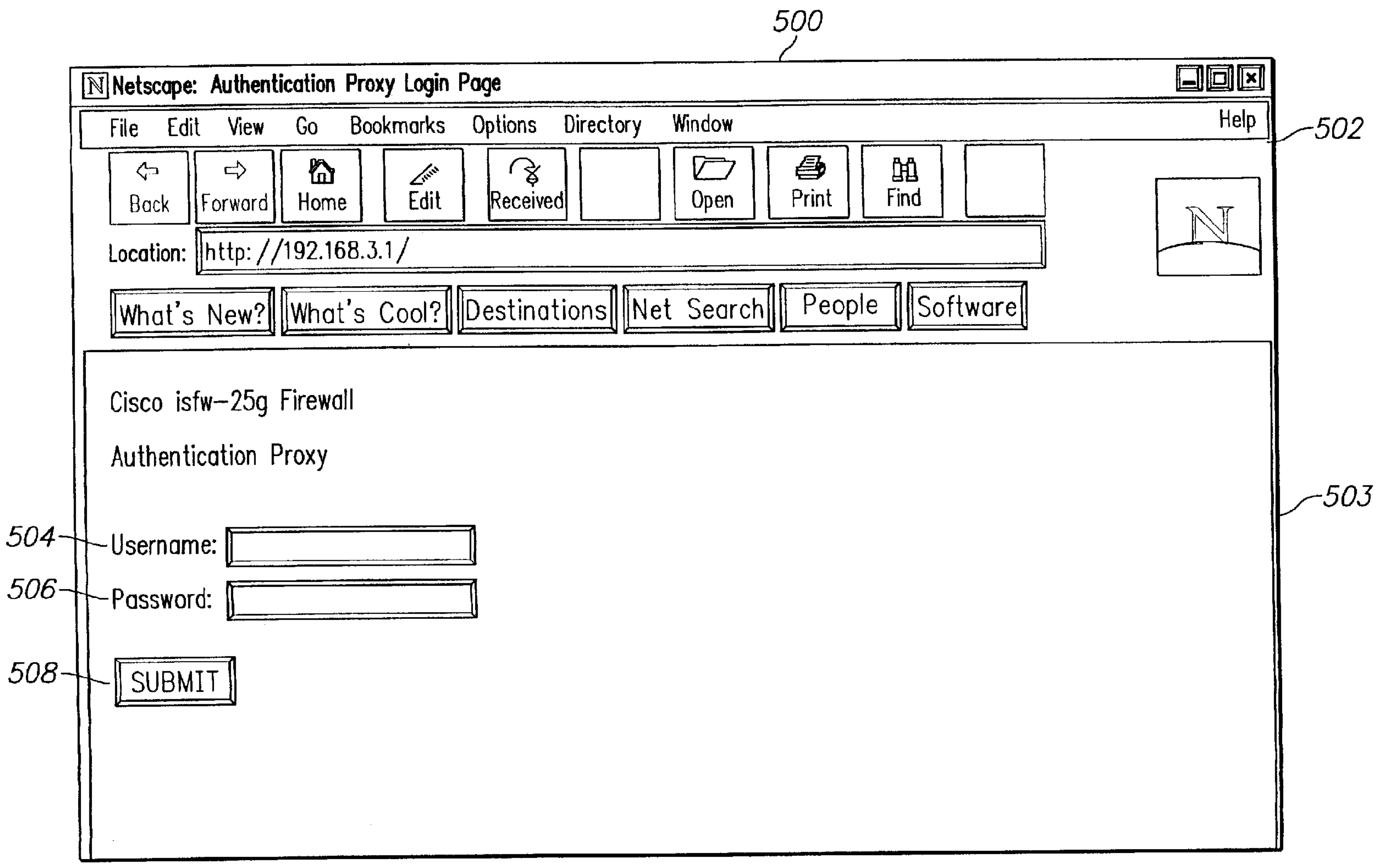

User Profiles
220
216 Intranet
Target Server 222
AA Server 218
406
405
428 In_ACL
Internal Interface 422
430 Out_ACL
Firewall Router 210
400 Auth Proxy
424 In_ACL
External Interface 420
426 Out_ACL
Authentication Cache 432
Authentication Cache 434
Authentication Cache 436
401 http
403 login
404 username/personal
408a success
408b reload
409
410
LAN 206
User 302
Browser 304
Client Host 306

FIG. 4



Netscape: Authentication Proxy Login Page

Location: http://192.168.3.1/

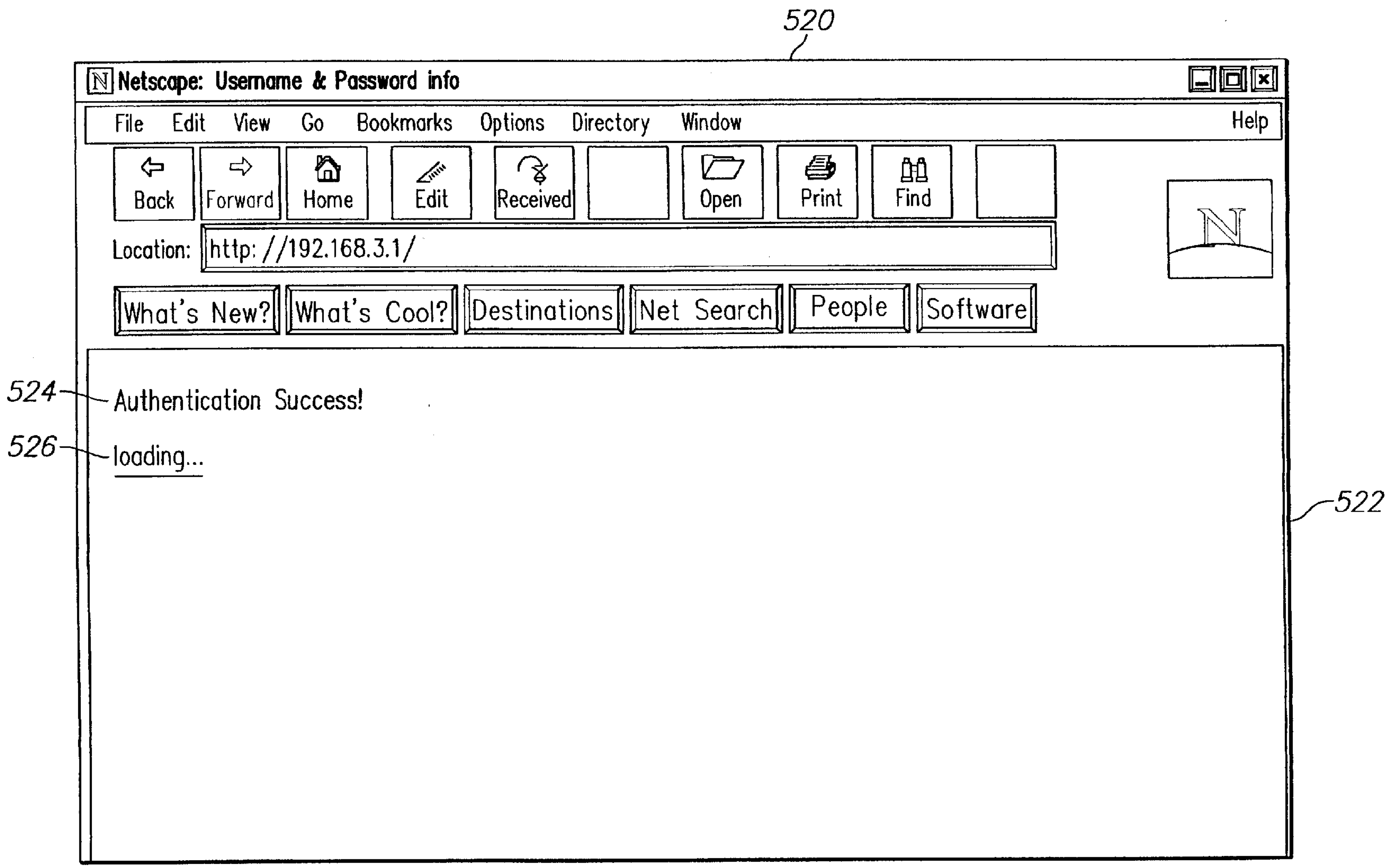
Cisco isfw-25g Firewall

Authentication Proxy

Username:

Password:

FIG. 5A



Location: http://192.168.3.1/

Authentication Success!

loading...

FIG. 5B

FIG. 6

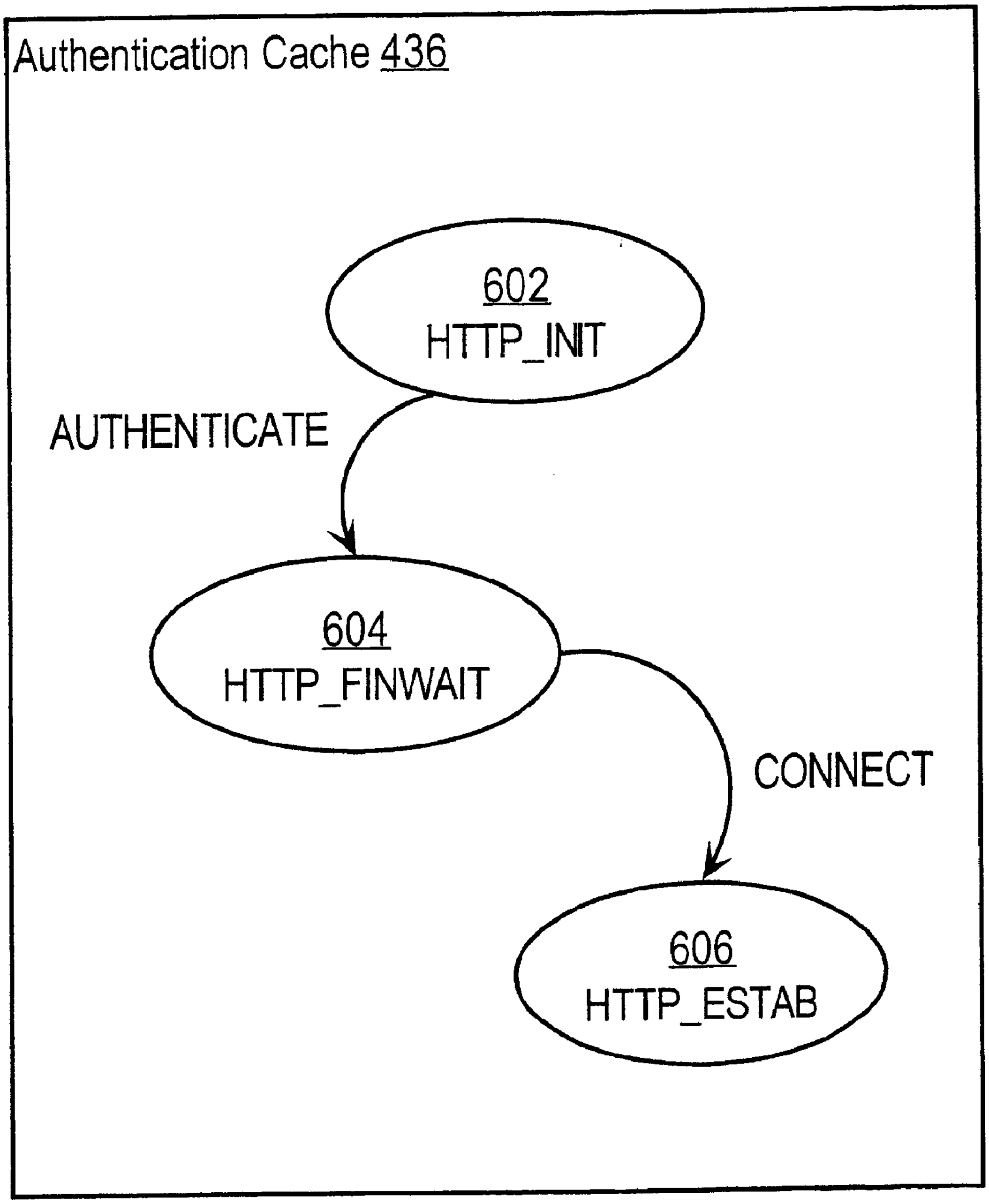

Authentication Cache 436
602
HTTP_INIT
AUTHENTICATE
604
HTTP_FINWAIT
CONNECT
606
HTTP_ESTAB

FIG. 7A

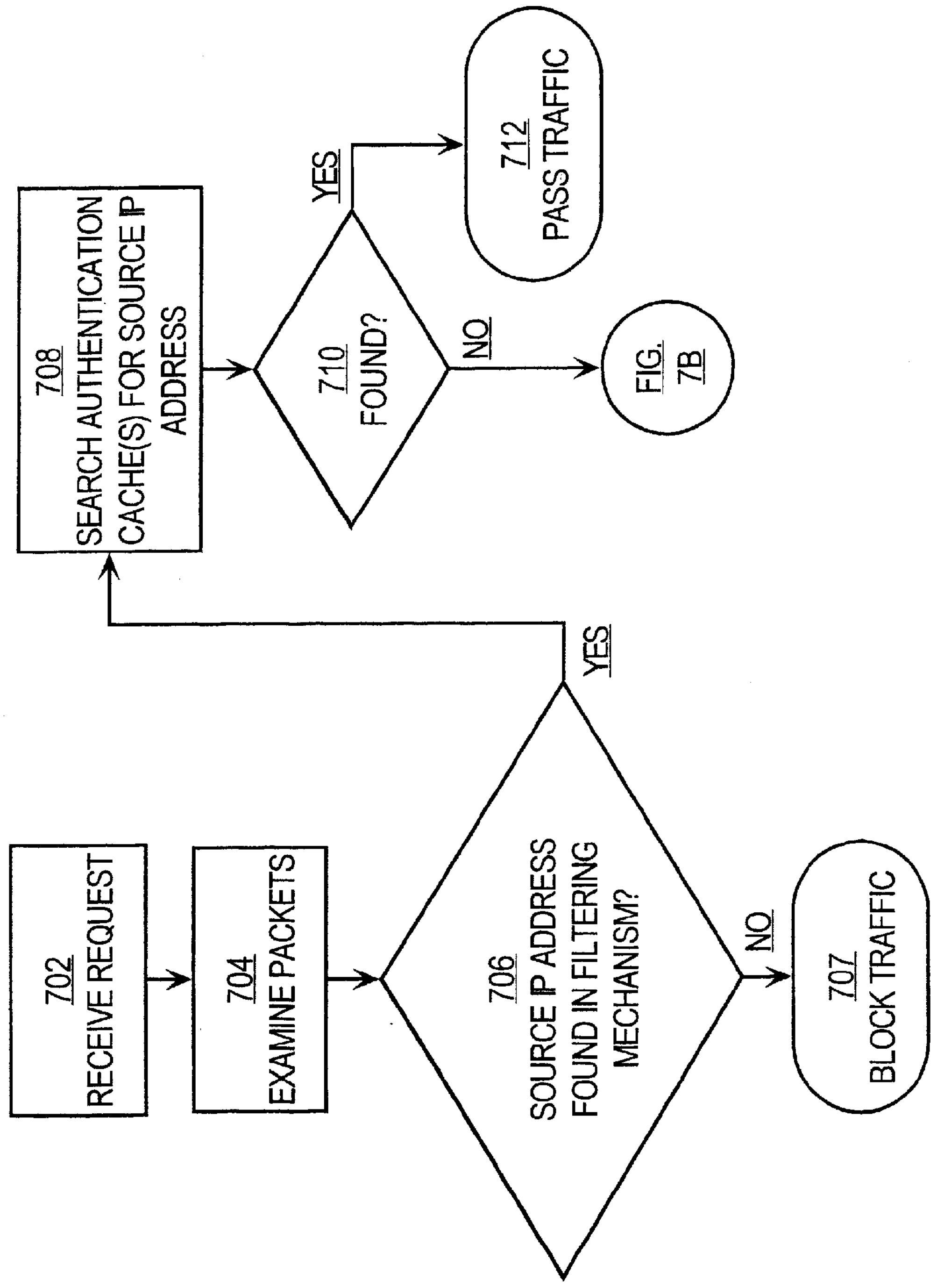
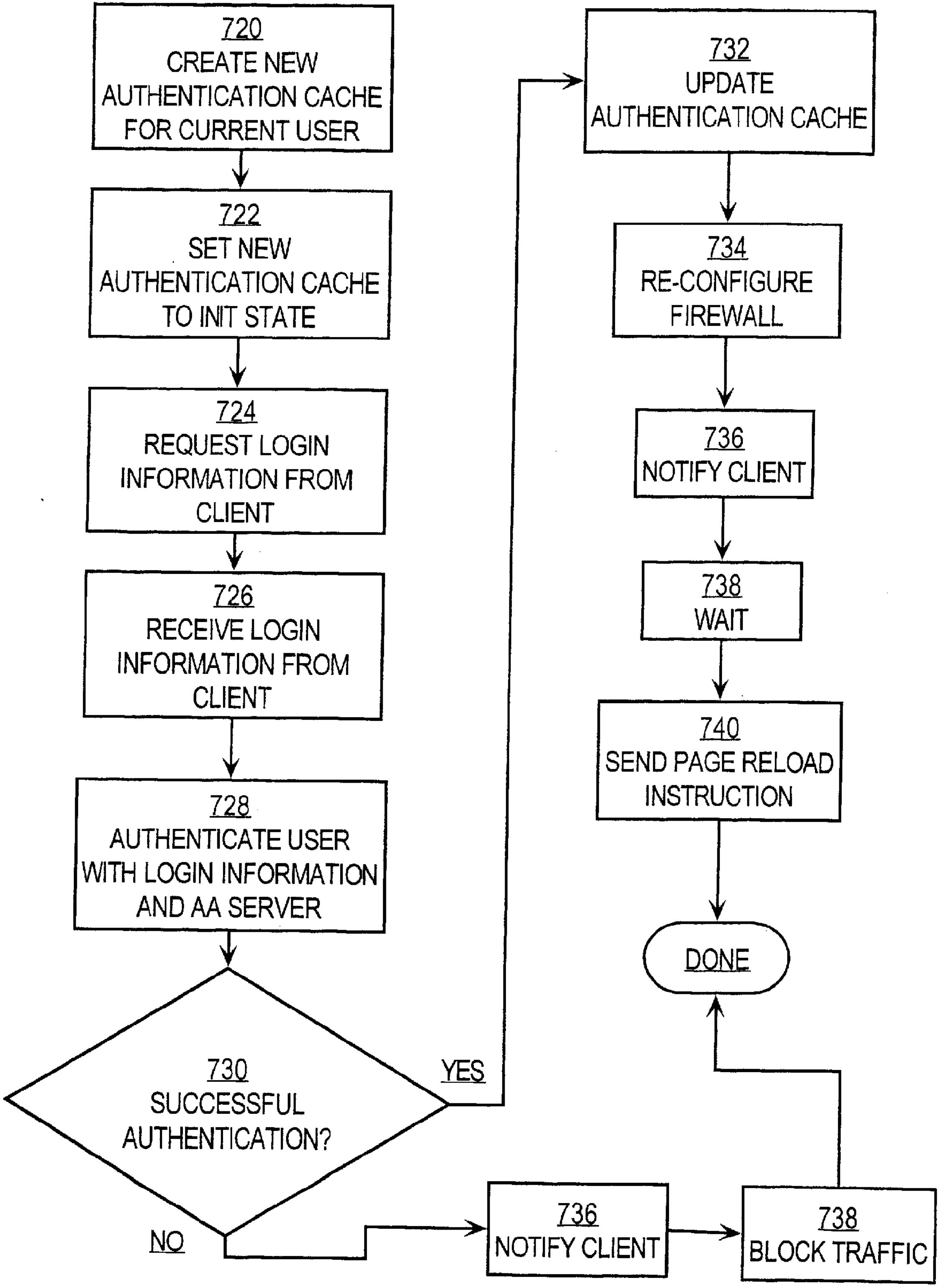

702
RECEIVE REQUEST
704
EXAMINE PACKETS
706
SOURCE IP ADDRESS FOUND IN FILTERING MECHANISM?
NO
707
BLOCK TRAFFIC
YES
708
SEARCH AUTHENTICATION CACHE(S) FOR SOURCE IP ADDRESS
710
FOUND?
YES
712
PASS TRAFFIC
NO
FIG. 7B

FIG. 7B


720
CREATE NEW AUTHENTICATION CACHE FOR CURRENT USER
722
SET NEW AUTHENTICATION CACHE TO INIT STATE
724
REQUEST LOGIN INFORMATION FROM CLIENT
726
RECEIVE LOGIN INFORMATION FROM CLIENT
728
AUTHENTICATE USER WITH LOGIN INFORMATION AND AA SERVER
730
SUCCESSFUL AUTHENTICATION?
YES
NO
732
UPDATE AUTHENTICATION CACHE
734
RE-CONFIGURE FIREWALL
736
NOTIFY CLIENT
738
WAIT
740
SEND PAGE RELOAD INSTRUCTION
DONE
736
NOTIFY CLIENT
738
BLOCK TRAFFIC

LOCAL AUTHENTICATION OF A CLIENT AT A NETWORK DEVICE

CROSS-REFERENCE TO RELATED APPLICATIONS; PRIORITY CLAIM

This application claims priority under 35 U.S.C. §120 as a Continuation of prior application Ser. No. 09/347,433, filed Jul. 2, 1999, now U.S. Pat. No. 6,463,474, the entire contents of which are hereby incorporated by reference as if fully set forth herein.

FIELD OF THE INVENTION

The present invention generally relates to management of computer networks, and relates more specifically to authentication and authorization mechanisms for network devices such as routers and firewalls.

BACKGROUND OF THE INVENTION

Computer users often access information, computer files, or other resources of computer networks from locations that are geographically or logically separate from the networks. This is referred to as remote access. For example, a user of a host or client that is part of a local area network ("LAN") may want to retrieve information that resides on a computer that is part of a remote network. Before a user can gain access to that computer, the user must first obtain permission to do so. In the interest of data integrity, and data confidentiality, many computer networks have implemented integrity and access control mechanisms to guard against unwanted network traffic or access by unauthorized users. On the other hand, a corporation may institute policies that restrict its employees from accessing certain web sites on the internet while using the corporation's computer resources. For example, Corporation C may disallow access to pornographic web sites. Corporation C's access control mechanism would prevent the employees from accessing such sites.

An example of an access control mechanism is a server that implements authentication, authorization, and accounting ("AAA") functions. Authentication is the process of verifying that the user who is attempting to gain access is authorized to access the network and is who he says he is. Generally, after authentication of a user, an authorization phase is carried out. Authorization is the process of defining what resources of the network an authenticated user can access.

Several authentication and authorization mechanisms are suitable for use with operating systems that are used by network devices, such as the Internetworking Operating System ("IOS") commercially available from Cisco Systems, Inc. However, most prior authentication and authorization mechanisms are associated with dial-up interfaces, which can create network security problems. In a dial-up configuration, a remote client uses a telephone line and modem to dial up a compatible modem that is coupled to a server of the network that the remote client wishes to access. In another dial-up configuration, a remote client first establishes a dial-up connection to a server associated with an Internet Service Provider, and that server then connects to the network server through the global, public, packet-switched internetwork known as the Internet. In this configuration, the network server is coupled directly or indirectly to the Internet.

Unfortunately, information requests and other traffic directed at a network server from the Internet is normally considered risky, untrusted traffic. An organization that owns or operates a network server can protect itself from unauthorized users or from unwanted traffic from the Internet by using a firewall. A firewall may comprise a router that executes a "packet filter" computer program. The packet filter can selectively prevent information packets from passing through the router, on a path from one network to another. The packet filter can be configured to specify which packets are permitted to pass through the router and which should be blocked. By placing a firewall on each external network connection, an organization can prevent unauthorized users from interfering with the organization's network of computers. Similarly, the firewall can be configured to prevent the users of the organization's network of computers from accessing certain undesirable web sites on the Internet.

One common method of remote access using the Internet is telnet, a protocol used to support remote login sessions that defines how local and remote computers talk to each other to support a remote login session. "Telnet" is also the name of a remote login program commonly used in networks based on Transmission Control Protocol/Internet Protocol ("TCP/IP"), a set of protocols that define how communications occur over the Internet. Past authentication and authorization mechanisms were produced to work with firewalls in the context of telnet. An example of an authentication and authorization mechanism that works with telnet is "Lock and Key" for IOS, commercially available from Cisco Systems, Inc.

However, a major drawback of telnet is that the client must know, before making any connection request, the Internet Protocol address ("IP address") of the firewall that is protecting the target network which the client is attempting to access. An IP address is a unique 32-bit binary number assigned to each firewall, router, host computer or other network element that communicates using IP. Obtaining the IP address of a firewall can be inconvenient or impractical because there are so many IP addresses currently assigned to network devices. Further, IP addresses normally are guarded closely by the network owner, because knowledge of an IP address enables unauthorized traffic to reach the device identified by the IP address.

Moreover, once a user successfully uses the authentication and authorization mechanism to secure a logical path through the firewall, the user may be restricted to one type of network traffic for the connection. For example, a firewall can be configured to provide a path through the firewall for a specific type of network traffic as specified by a user profile that is associated with each authenticated user. The user profile contains information on what the user is authorized to do on the network. The user profile may specify, for example, that the user may use only File Transfer Protocol ("FTP") traffic. Thus, the user may use the path through the firewall only for FTP traffic, for the duration of that connection. Furthermore, the user profile associated with the user contains a specific IP address that specifies the host or client from which the user can attempt to secure a logical path through the firewall. Thus, a user is not free to use any one of several computers that may be available to access the target network. Also, the user may not be free to use a client in a network that employs Dynamic Host Configuration Protocol (DHCP). DHCP assigns dynamic IP addresses to the devices on a network. Thus, a client in a DHCP environment can have a different IP address every time it connects to the network.

Based on the foregoing, there is a clear need for a mechanism allowing users to use remote access via the Internet without requiring advance knowledge of the IP address of the firewall router, and without restricting a user to a particular host or client.

In particular, there is a need for an authentication and authorization mechanism in the context of remote access via the Internet that does not rely on telnet and that allows the passage of different types of traffic for a given connection.

SUMMARY OF THE INVENTION

The foregoing needs, and other needs and objects that will become apparent for the following description, are achieved in the present invention, which comprises, in one aspect, a method of controlling access of a client to a network resource using a network device that is logically interposed between the client and the network resource, the method comprising creating and storing client authorization information at the network device, wherein the client authorization information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource; receiving a request from the client to communicate with the network resource; determining, at the network device, whether the client is authorized to communicate with the network resource based on the authorization information; and reconfiguring the network device to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information.

One feature of this aspect is that creating and storing client authorization information comprises the steps of creating and storing in the network device a set of authorization information for each client that communicates with the network device. According to another feature of this aspect is that creating and storing client authorization information comprises the steps of creating and storing in the network device an authentication cache for each client that communicates with the network device. In another feature, creating and storing client authorization information comprises the steps of creating and storing in the network device a plurality of authentication caches, each authentication cache uniquely associated with one of a plurality of clients that communicate with the network device, each authentication cache comprising information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource

According to still another feature, determining whether the client is authorized to communicate with the network resource comprises the step of determining whether information in the request identifying the client matches information in a filtering mechanism of the network device and the authorization information stored in the network device.

In another feature, determining whether the client is authorized to communicate with the network resource comprises the steps of: determining whether a source IP address of the client in the request matches information in a filtering mechanism of the network device; and if so, determining whether the source IP address matches the authorization information stored in the network device.

In another feature, determining whether the client is authorized to communicate with the network resource comprises the steps of: determining whether a source IP address of the client in the request matches information in an a filtering mechanism of the network device; determining whether the source IP address matches the authorization information stored in the network device; and when the source IP address fails to match the authorization information stored in the network device, determining if user identifying information received from the client matches a profile associated with the user that is stored in an authentication server that is coupled to the network device.

In another feature, determining whether the client is authorized to communicate with the network resource comprises the steps of: determining whether client identifying information in the request matches information in a filtering mechanism of the network device; determining whether the client identifying information matches the authorization information stored in the network device; and only when the client identifying information fails to match the authorization information stored in the network device, then: creating and storing new authorization information in the network device that is uniquely associated with the client; requesting login information from the client; authenticating the login information by communicating with an authentication server that is coupled to the network device; and updating the new authorization information based on information received from the authentication server.

According to another feature, requesting login information from the client comprises sending a Hypertext Markup Language login form to the client to solicit a username and a user password; and authenticating the login information by communicating with an authentication server that is coupled to the network device comprises determining, from a profile associated with a user of the client stored in the authentication server, whether the username and password are valid.

In another feature, the method further comprises the steps of: creating and storing an inactivity timer for each authentication cache, wherein the inactivity timer expires when no communications are directed from the client to the network resource through the network device during a predetermined period of time; removing the updated authentication information when the inactivity timer expires.

In another feature, determining whether the client is authorized to communicate with the network resource comprises the steps of: determining whether a source IP address in the request matches information in a filtering mechanism of the network device; determining whether the source IP address matches the authorization information stored in the network device; and only when the source IP address fails to match the authorization information stored in the network device, then: creating and storing in the network device a new authentication cache that is uniquely associated with the client; requesting login information from the client; authenticating the login information by communicating with an authentication server that is coupled to the network device; and updating the new authentication cache based on information received from the authentication server.

According to another feature, reconfiguring the network device comprises the steps of creating and storing one or more commands to the network device whereby one or more interfaces of the network device are modified to permit communications between the client and the network resource.

In another feature, the method further involves instructing the client to reload the network resource that was identified in the request from the client when it is determined that the client is authorized to communicate with the network resource.

According to another feature, the method further comprises the steps of waiting a pre-determined period of time, and instructing the client to reload the network resource that was identified in the request from the client when it is determined that the client is authorized to communicate with the network resource.

In another feature, the network device comprising a firewall that protects the network resource by selectively blocking messages initiated by client and directed to the network resource, the firewall comprising an external interface and an internal interface, the firewall comprising an Output Access Control List at the internal interface and an Input Access Control List at the external interface, wherein reconfiguring the network device comprises the step of: substituting the IP address in a user profile information associated with a user of the client to create a new user profile information, wherein the user profile associated with the user of the client is received from an authentication server that is coupled to the network device; and adding the new user profile information as temporary entries to the Input Access Control List at the external interface and to the Output Access Control List at the internal interface.

According to still another feature, the method further involves: creating and storing an inactivity timer for the authorization information, wherein the inactivity timer expires when no communications are directed from the client to the network resource through the network device during a pre-determined period of time; associating the temporary entries with the authorization information and the client; and removing the temporary entries and the authorization information from the network device if the inactivity timer expires.

In another feature, the authorization information includes a table of hashed entries and wherein associating the temporary entries to the authorization information further comprises storing the temporary entries in the table of hashed entries.

In another feature, the network device comprising a firewall that protects the network resource by selectively blocking messages initiated by client and directed to the network resource, the firewall comprising an external interface and an internal interface, the firewall comprising an Output Access Control List at the external interface and an Input Access Control List at the internal interface, wherein reconfiguring the network device comprises the step of: substituting the IP address in a user profile information associated with a user of the client to create a new user profile information, wherein the user profile associated with the user of the client is received from an authentication server that is coupled to the network device; and adding the new user profile information as temporary entries to the Input Access Control List at the internal interface and to the Output Access Control List at the external interface.

In another feature, the method further involves: creating and storing an inactivity timer for the authorization information, wherein the inactivity timer expires when no communications are directed from the client to the network resource through the network device during a predetermined period of time; associating the temporary entries with the authorization information and the client; and removing the temporary entries and the authorization information from the network device if the inactivity timer expires.

In another feature, the authorization information includes a table of hashed entries and wherein associating the temporary entries to the authorization information further comprises storing the temporary entries in the table of hashed entries.

According to another aspect, the invention encompasses computer system for controlling access of a client to a network resource using a network device that is logically interposed between the client and the network resource, comprising: one or more processors; a storage medium carrying one or more sequences of one or more instructions including instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of: creating and storing client authorization information at the network device, wherein the client authorization information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource; receiving a request from the client to communicate with the network resource; determining, at the network device, whether the client is authorized to communicate with the network resource based on the authorization information; and reconfiguring the network device to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information.

According to another aspect, the invention involves a router that is logically interposed between a client and a network resource and that controls access of the client to the network resource, comprising: one or more processors; a storage medium carrying one or more sequences of one or more instructions including instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of: creating and storing client authorization information at the router, wherein the client authentication information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource; receiving a request from the client to communicate with the network resource; determining, at the router, whether the client is authorized to communicate with the network resource based on the authorization information; and reconfiguring the router to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information.

In other aspects, the invention encompasses a computer apparatus, a computer readable medium, and a carrier wave configured to carry out the foregoing steps.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

FIG. **1** is a block diagram that illustrates a computer system upon which an embodiment may be implemented;

FIG. **2** is a block diagram of a system providing an authentication proxy in a network environment;

FIG. **3** is a block diagram of the system in FIG. **2** showing certain internal details;

FIG. **4** is a block diagram of the system in FIG. **3** showing certain paths of network traffic;

FIG. **5**A illustrates a display of a graphical user interface containing a dialog box for soliciting a username and password;

FIG. **5**B illustrates a display of the graphical user interface informing of an authentication success;

FIG. **6** is a state diagram of states in which an authentication cache may execute;

FIG. 7A is a flow diagram of a process of proxy authentication;

FIG. 7B is a flow diagram of further steps in the process of FIG. 7A.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A method and apparatus for authentication and authorization proxy mechanisms for firewalls that protect networks is described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

Operational Context

The present invention may be implemented using various client protocols such as Telnet, File Transfer Protocol (FTP), or HyperText Transfer Protocol (HTTP). For purposes of illustration, the invention is described in the context of an HTTP client protocol.

In one embodiment, a user of a client that is part of a local area network ("LAN") attempts to remotely access a server ("target server") or some other resource, such as a peer client or device. The target server and or peer are part of a packet-switched private network that operates using TCP/IP and other Internet standards ("intranet"). The client is connected to the Internet through the LAN, and the intranet is also connected to the Internet. Alternatively, the client may be a stand-alone computer connected to the Internet through a dial-up connection or a digital communication service such as an Integrated Services Digital Network (ISDN) connection. In another embodiment, a user of a client from within the intranet attempts to access a target server or other resource that is not part of the same intranet as that of the client.

When the target server executes an HTTP server, the client can remotely access the target server over the Internet by using a Web browser to specify a Web page on the target server. Using a Web browser to specify a Web page is hereafter referred to as an "HTTP request" or as "transmitting HTTP packets." A Web page of the target server may be accessed using identifying information, such as a Uniform Resource Locator ("URL") and therefore the Web page is sometimes called the "target URL."

The HTTP packets are intercepted by a firewall that protects the intranet from unwanted network traffic originating from the Internet (inbound traffic) and can prevent users of clients from within the intranet from accessing undesirable web sites on the Internet (outbound traffic). For purposes of illustration, use of an embodiment with inbound traffic is described in further detail below.

Upon intercepting the HTTP packets, the firewall requests, from the client, authentication information such as username and password. In response to receiving the authentication information, the firewall performs an authentication and authorization process. If the username is successfully authenticated, then the firewall is dynamically configured to open a passageway for the HTTP packets as well as other types of network traffic initiated from the user on the client. The other types of network traffic that are permitted through the passageway are specified in a user profile for that particular user. In this context, "open a passageway" means that the firewall re-configures itself, in response to successful authentication, so that packets that would otherwise be barred are now allowed to pass.

In this configuration, the firewall provides an authentication and authorization mechanism that substitutes for an authentication and authorization mechanism elsewhere in the network. Accordingly, the mechanism described in this document is referred to as an "Authentication Proxy." The Authentication Proxy may comprise one or more software components, executed by a router. In one embodiment, the Authentication Proxy can be enabled on a router interface to intercept traffic initiated from a client that is not yet authenticated. The Authentication Proxy is responsible for validating the user associated with the client and for applying the appropriate user profile to the router interface. The authentication and authorization process and the dynamic configuration of the firewall are described in further detail below.

FIG. **2** is a block diagram of a system **200** in which an embodiment of an Authentication Proxy can be used. Generally, system **200** includes a LAN **206**, and a local, packet-switched network that uses Internet protocols, or intranet, **216**. The LAN **206** and the intranet are both connected to a global network such as the Internet. The LAN **206** and intranet **216** are respectively located in logically distinct regions, such as first region **202** and second region **204**, which may be geographically separate. A firewall router **210** is logically interposed between LAN **206** and the intranet **216**.

LAN **206** is a local area network comprising any number of network devices **208***a*, **208***b*, **208***c* interconnected by one or more communications channels **209**. Ethernet, Token Ring, other protocols can characterize the communications channels **209**.

Firewall router **210** is a specialized router that carries out firewall functions. The firewall router **210** is coupled to intranet **216**, and an authentication and authorization server **218** ("AAA server"). The firewall router **210** controls remote access to intranet **216**. AAA server **218** is a computer, or a group of hardware or software components or processes that cooperate or execute in one or more computer systems. The AAA server **218** has access to a database **220** that stores authentication and authorization information on users ("user profile"). The firewall router **210** cooperates with the AAA server **218** to perform authentication and authorization services. In this way, firewall router **210** restricts and controls access of traffic originating from intranet **216** and directed to LAN **206**.

Intranet **216** is one or more interconnected networks ("internetwork") each of which may comprise any number of network devices. A target server **222** is part of Intranet **216** and is a computer system that may be remotely accessed by a client on LAN **206**. In certain embodiments, the AAA server **218** and database **220** may be coupled indirectly to firewall router **210** through the Internet. In this configuration, the AAA server **218** and database **220** are said to be one or more "hops" removed from the firewall router **210**. A hop is the next place in the network to send a packet so that a packet will eventually reach its destination.

FIG. **3** is a block diagram showing certain internal details of the system in FIG. **2**. In this example, one of the network devices **208***a*–**208***c* of LAN **206** is a client **306**, such as a personal computer or workstation. Client **306** is associated with a user **302**. In certain embodiments, client **306** is configured with or coupled to multiple modems or ISDN bearer channels that can be used to establish one or more connections **308, 310** from client **306** to firewall router **210**. In one embodiment, client **306** runs a browser **304**, which is an application program used to retrieve and display Web pages or other information stored in target server **222**. Commercial examples of browser **304** include Netscape Navigator® or Microsoft Internet Explorer®. User **302** can use browser **304** to send an HTTP request over LAN **206** and Intranet **216** to target server **222**.

Authentication and Authorization

FIG. **4** is a block diagram of the system of FIG. **3** that illustrates providing an Authentication Proxy.

FIG. **7**A and FIG. **7**B are flow diagrams that illustrate a method for carrying out proxy authentication at a router. As an example, the method of FIG. **7**A and FIG. **7**B is described below in the context of the system of FIG. **4**. However, the method is not limited to this context.

As in FIG. **3**, the system of FIG. **4** includes User **302** who is associated with Client **306**. A Browser **304** is executed by Client **306**, which is part of LAN **206**. Client **206** communicates with firewall router **210** over LAN **206** using messages illustrated by paths **401, 403, 404, 408***a*, **408***b*. Firewall router **210** is coupled to client **306**, to AAA Server **218**, and to intranet **216**. One function of Client **306** is to request and retrieve electronic documents, applications or services that are available at target server **222**.

A filtering mechanism **219** is part of the configuration of Authentication Proxy **400**. The filtering mechanism **219** contains information identifying one or more IP addresses of clients that are authenticated in the network to which the firewall **210** belongs.

Paths of network traffic are depicted by solid lines or dotted lines with arrowheads. Paths **401, 403, 404, 408***a*, **408***b* illustrate network traffic communicated between client **306** and firewall router **210**. Paths **405** and **406** illustrate network traffic communicated between firewall router **210** and AAA server **218**. Paths **409** and **410** illustrate network traffic communicated between the client **306** and target server **222**. Each path represents one or more packets, messages or sessions communicated among the respective end elements. Paths **409** and **410** pass through firewall router **210** but do not stop within the firewall router.

Firewall router **210** has an external interface **420** and an internal interface **422**. Each interface **420, 422** defines how firewall router **210** regulates the flow of traffic arriving at or sent from the respective interface. The external interface **420** includes an input Access Control List ("ACL") **424**, and an output ACL **426**. Internal interface **422** also includes an input ACL **428** and an output ACL **430**.

Access control lists filter packets and can prevent certain packets from entering or exiting a network. Each ACL is a list of information that firewall router **210** may use to determine whether packets arriving at or sent from a particular interface may be communicated within or outside the firewall router. For example, in an embodiment, input ACL **424** may comprise a list of IP addresses and types of allowable client protocols. Assume that firewall router **210** receives an inbound packet from client **306** at external interface **420** that is intended for target server **222**. If the IP address of client **306** is not stored in input ACL **424**, then firewall router **210** will not forward the packet further within the circuitry or software of the firewall router. Output ACL **426** similarly controls the delivery of packets from firewall router **210** to resources located outside external interface **420**. Input ACL **428** and output ACL **430** govern packet flow to or from internal interface **422**.

The firewall router **210** also includes any number of authentication caches **432, 434**. The access control lists are linked to the authentication caches. Each authentication cache represents a valid user authentication. Each authentication cache may include a table of hashed entries of information such as a source IP address, a destination IP address, a source port value, a destination port value, and state information.

Firewall router **210** also includes Authentication Proxy **400**. In one embodiment, Authentication Proxy **400** is one or more software elements, modules or processes executed by firewall router **210** and coupled to the external interface **420**, internal interface **422**, and authentication caches **432, 434**. Authentication Proxy **400** may be configured to carry out the functions described in this document.

Authentication

In an embodiment, Authentication Proxy **400** is activated or enabled at the firewall router **210**. Authentication Proxy **400** may be implemented as a software process within firewall router **210** that may be accessed using commands in a standard command-line router programming language.

For purposes of illustrating an example of operation of authentication proxy **400**, assume that Authentication Proxy **400** receives a request for something in a network that is protected by the Authentication Proxy, as shown by block **702** of FIG. **7**A. For example, User **302** uses browser **304** to send an HTTP request from client **306** for an electronic document, application or resource available at target server **222**. User **302** is not authenticated in intranet **216**. The HTTP request travels along path **401** from client **306** to firewall router **210**. Each packet of an HTTP request includes a header portion that contains one or more fields of information. The fields include, among other things, values for source IP address and destination IP address of that packet.

In block **704**, packets of the request are examined. For example, when the HTTP request arrives at the external interface **420** of the firewall router **210**, Authentication Proxy **400** examines packets of the request. In block **706**, the process determines whether a source IP address of the request is found in the standard access control list. For example, Authentication Proxy **400** determines whether the source IP address in the header field of the packets corresponds to any entry in the filtering mechanism **219** configured in the Authentication Proxy **400**.

If the test of block **706** is affirmative, then control passes to block **708** in which the authentication caches are searched for the source IP address. In block **710**, the process tests whether the source IP address is found. For example, if Authentication Proxy **400** determines that the source IP address matches at least one IP address stored in the filtering mechanism **219**, then the Authentication Proxy **400** attempts to authenticate the user **302**. In the preferred embodiment, Authentication Proxy **400** searches authentication caches **432, 434** for the source IP address. The goal of this search is to determine if the source IP address of the HTTP packet corresponds to an entry in any of the authentication caches **432, 434**.

Assume that the filtering mechanism **219** contains a source IP address that matches the IP address of client **306**, so that the test of block **706** is affirmative. However, User **302** is not yet authenticated, so that the test of block **710** is negative. Thus, the authentication caches have no hashed entries that match the source IP address found in the header field of the HTTP packet. As a result, without further action the firewall router **210** will intercept and will not forward the

HTTP packet, instead of providing it with a logical passageway through the router.

In one embodiment, at this stage, control is passed to block **720** of FIG. **7**B. Preferably, a new authentication cache **436** is created for User **302**, as shown in block **720**. Each authentication cache may operate in a plurality of states. FIG. **6** is a state diagram that illustrates the states in which an authentication cache may operate. Initially, the state of the new authentication cache **436** is set to the HTTP_Init state **602**, as shown by block **722**. The HTTP_Init state **602** indicates that the authentication and authorization state for User **302** is at an initial state. In one embodiment, an HTTP packet is provided with a logical passageway through the firewall router **210** only when the state of an authentication cache that contains information about User **302** is set to the HTTP_Estab state **606**, which is described further below.

If the source IP address of the HTTP packet from client **306** does not match any of the entries in the filtering mechanism **219**, then Authentication Proxy **400** denies passage to the HTTP packet and makes no attempt at authentication, as shown by block **707** of FIG. **7**A. As a result, advantageously, the packet is turned away at the interface and never reaches internal software and hardware elements of the firewall router.

If Authentication Proxy **400** is not configured with the filtering mechanism **219**, then Authentication Proxy **400** will intercept traffic, for purposes of authentication, from all hosts whose connection initiating packets arrive at a firewall interface for which Authentication Proxy **400** has been enabled. It is not practical or desirable to attempt authentication for every packet that arrives at the firewall. For example, there is no point in authenticating packets that arrive from sites known to be hostile or sites of an unknown nature. Thus, in one embodiment, a filtering mechanism is defined to enable the Authentication Proxy **400** to bypass traffic from such sites.

Referring again to FIG. **7**B, after the new authentication cache is created, login information is requested from the client, as shown in block **724**. For example, Authentication Proxy **400** obtains authentication information from User **302** by sending a login form to client **306**. The login form is an electronic document that requests User **302** to enter username and password information, as shown by path **403**.

FIG. **5**A is an example of a graphical user interface ("GUI") representation of the login form **500** that may be sent by Authentication Proxy **400**. The form **500** may be displayed by browser **304**, as indicated by command area **502**. Form **500** also includes a data entry pane **503** having a Username field **504**, and a Password field **506**. A User **302** may enter username information in Username field **504** and may enter password information in Password field **506**. To communicate the username and password information to Authentication Proxy **400**, the user selects a "Submit" button **508**. In response, the login information is communicated to firewall router **210**, as indicated by path **404**.

The login form may be composed using the HyperText Markup Language ("HTML") format. Table 1 presents source code for an appropriate HTML document that may be used for the login form of FIG. **5**A.

TABLE 1

EXAMPLE LOGIN FORM

```
<HTML>
  <HEAD>
   <TITLE>
     Authentication Proxy Login Page
   </TITLE>
  </HEAD>
  <BODY BGCOLOR=#FFFFFF>
   <H1>Cisco "router name" Firewall<H1>
   <H2>Authentication Proxy<H2><BR><BR><P><P><P>
   <FORM METHOD=POST ACTION=\"\">
   <INPUT TYPE=HIDDEN NAME=TIMETAG VALUE=>
    Username: <INPUT TYPE=TEXT NAME=UNAME><BR><BR>
    Password: <INPUT TYPE=PASSWORD NAME=PWD><BR><BR>
   <INPUT TYPE=SUBMIT NAME=SUBMIT VALUE=SUBMIT>
   </FORM>
  </BODY>
</HTML>
```

Login information is received from the client, as shown by block **726**. Block **726** may involve receiving username and password information in an HTTP message that is generated when a user fills in and submits the form of Table 1. In block **728**, the user is authenticated using the login information and the AAA server **218**. For example, upon receipt of the username and password information, Authentication Proxy **400** attempts to authenticate the user by sending the username and password to the AAA server **218**, as shown by path **405**. AAA server **218** has access to database **220**, which contains user profiles of authorized users.

In block **730**, the process tests whether authentication is successful. Successful authentication will occur when AAA server **218** verifies that the username and password information are recognized and correct. If Authentication Proxy **400** successfully authenticates User **302** using AAA Server **218**, then AAA server **218** informs firewall router **210** of the successful authentication, as shown by path **406**. In block **732**, the process updates the current authentication cache with the source IP address of the client and with information contained in the user profile of User **302**. In block **734**, the firewall is re-configured to allow packets using protocols specified in user's profile of User **302** and associated with that IP address to pass through the firewall freely. Block **734** may involve creating and executing one or more router commands. In this way, subsequent authentication attempts against the authentication cache at the firewall will succeed, precluding the need to connect to AAA Server **218** repeatedly. This provides a significant advantage and improvement in authentication speed and efficiency.

Authentication Proxy **400** then informs Client **306** that authentication succeeded. As shown in block **736**, the client is notified. In the preferred embodiment, Authentication Proxy **400** sends an HTML page containing a success message, as shown by path **408***a*.

FIG. **5**B is an example of an HTML page that Authentication Proxy **400** may send to Client **306**. Pane **522** of window **520** contains an "Authentication Successful" message **524**.

Pane **522** also includes a second message **526** informing Client **306** that further action is being taken. For example, the second message **526** may inform Client **306** that the resource originally requested from target server **222** is now being loaded or accessed. In the preferred embodiment, the original HTTP request is automatically forwarded by firewall router **210** to target server **222** without further intervention by Client **306** or User **302**. Thus, Authentication Proxy **400** automatically interrupts and resumes the request.

As shown in block **738**, the process waits. For example, after Authentication Proxy **400** sends the "Authentication Success" message **524** to User **302**, the Authentication Proxy enters a wait state for a short, pre-determined period of time. During the wait state, a short period of time is allowed to elapse to enable client **306** and firewall router **210** to communicate handshaking messages and carry out related processing associated with establishing an HTTP connection. The delay period also allows the firewall router enough time to execute any commands that are issued as part of block **734**. In one embodiment, a period of three (3) seconds elapses.

In block **740**, the process sends a page reload instruction to the client. For example, after the delay period, Authentication Proxy **400** sends one or more messages to Client **306** that instruct the client to reload the target URL, as indicated by path **408***b* of FIG. **4**. In response, Client **306** re-issues the original HTTP request for a resource on target server **222**. By the time that the re-issued HTTP request arrives from Client **306** at firewall router **210**, the firewall router has been re-configured to provide an logical path **409** circumscribed only by the user profile of User **302** for the connection to the target server using the process described further below. Similarly, return traffic may travel from target server **222** to Client **306** on path **410** subject to the authorization information of the user profile of User **302**.

If the authentication is not successful, as shown in block **736** and block **738**, the process may notify the client with an appropriate message or page, and block traffic from the source IP address. Alternatively, Authentication Proxy **400** may permit Client **306** to re-try authentication a predetermined number of times.

Creating a Path Through the Firewall Using Dynamic Access Control List

When User **302** is authenticated successfully, Authentication Proxy **400** causes a user profile of User **302** to be downloaded from the AAA server **218** to the firewall router **210**. Authentication Proxy **400** uses the user profile to dynamically configure the external interface **420** and internal interface **422** of firewall router **210** to provide a passageway for network traffic that initiates from User **302** on client **306**. In this way, the types of traffic as specified by the user profile of User **302** and initiating from User **302** in the current session will pass freely through firewall router **210** to its destination.

In one embodiment, an authorized passageway through firewall router **210** is created using dynamic access control lists. In this embodiment, one or more entries may be added dynamically to each access control list **424**, **426**, **428**, **430** of external interface **420** and internal interface **422**.

When Authentication Proxy **400** sends a successful login confirmation message to User **302**, a user profile of User **302** is downloaded from AAA server **218**. In the preferred embodiment, the user profile is stored and retrieved in the form of one or more text commands, called proxy-access-list commands. Authentication Proxy **400** parses the proxy-access-list commands. Table 2 below contains samples of proxy-access-list commands. As each command is parsed, Authentication Proxy **400** replaces the source IP address field of the command with the IP address of the client **306**, to result in a modified proxy-access-list command. Each modified proxy-access-list command is added as a temporary entry to the access control lists at the external interface **420** and internal interface **422**. Specifically, proxy-access-list commands are added as temporary entries to the input ACL **424** of the router's external interface **420** and to the output ACL **430** of the router's internal interface **422**. Thus, access control lists **424**, **426**, **428**, **430** will grow and shrink dynamically as entries are added and deleted.

When the temporary entries have been added to the ACL at the appropriate interfaces of the firewall router **210**, the state of the current authentication cache is set to the HTTP_FINWAIT state **604**. The HTTP_FINWAIT state **604** indicates that the authentication cache is associated with an authenticated user, but that its associated client is not yet connected to a target server.

TABLE 2

EXAMPLE PROXY-ACCESS-LIST COMMANDS
permit tcp host 192.168.25.215 any eq telnet
permit tcp host 192.168.25.215 any eq ftp
permit tcp host 192.168.25.215 any eq ftp-data
permit tcp host 192.168.25.215 any eq smtp
deny tcp any any eq telnet
deny udp any any
permit tcp any any (76 matches)
permit ip any any

Authentication Cache Inactivity Timers

Each authentication cache may have an inactivity timer. In the preferred embodiment, an inactivity cache is a software process that is maintained for each authentication cache based on the amount of traffic coming through firewall router **210** from Client **306**. If the amount of traffic through the router for a particular client falls below a pre-determined threshold over a pre-determined period of time, then Authentication Proxy **400** generates an idle timeout event.

When an idle timeout event is generated for a particular client, Authentication Proxy **400** deletes the authentication cache associated with that client, and deletes all temporary entries in the access control lists that are associated with that particular client. This approach saves memory and ensures that the ACLs are regularly pruned.

For example, assume that the pre-determined inactivity timer value is 60 minutes. If Client **306** does not initiate any network traffic through the firewall router **210** for 60 minutes or more, then any subsequent traffic initiated from client **306** will be denied passage through the firewall router **210**. In one embodiment, Authentication Proxy **400** will prompt User **302** to renew authentication for a new HTTP connection.

Preferably, the temporary entries in the ACLs are not automatically deleted when a user terminates a session. Instead, the temporary entries remain stored until the configured timeout is reached, or until they are specifically deleted by the system administrator. Using this process, the user is not required to log in a second time in the event of an inadvertent or transient disconnection, but unused entries are removed from the ACLs when a true idle condition occurs.

In addition, each temporary entry to an ACL **424**, **426**, **428**, **430** is linked to an associated authentication cache **432**, **434**. Preferably, a temporary entry of an ACL is linked to an associated authentication cache by hashing the temporary entries and storing the hashed entries in the hash table that is maintained by the authentication cache.

Connection and Communication with Target Server

When the interfaces of firewall router **210** have been configured with the new temporary entries in the ACLs, the result is that a logical passageway is opened through the firewall to allow certain types of traffic specified in the user profile of User **302** and initiating from Client **306** to pass unobstructed to target server **222**. Using standard HTTP request-response communications, Client **306** establishes an HTTP connection to target server **222**. When the connection to target server **222** is established, Authentication Proxy **400** changes the state of the authentication cache associated with the User **302** to the HTTP_ESTAB state **606**.

As long as an idle timeout event does not occur, an authentication cache associated with a client remains valid. As long as the authentication cache associated with a client remains valid, the firewall router **210** does not attempt to intercept any subsequent traffic initiating from that client.

For example, assume that Client **306** continues to initiate network traffic or packets for passage through the firewall router **210** at least once every 60 minutes. Thus, the authentication cache **436** associated with client **306** remains valid and is not removed. Corresponding entries in the ACLs **424, 426, 428, 430** also remain valid. When a packet arrives at the firewall router **210**, Authentication Proxy **400** determines that the source IP address of the packet matches an entry in the authentication cache **436**. Accordingly, Authentication Proxy **400** will allow the packet to pass through firewall router **210**.

Moreover, subsequent traffic initiating from client **306** may travel to any destination, as specified by the temporary entries to the dynamic ACLs that are associated with client **306**. When a packet from client **306** arrives at firewall router **210**, Authentication Proxy **400** checks the hashed entries in the authentication cache **436** for a match in the destination IP address. If a match is found, then the Authentication Proxy **400** will allow the packet to pass through firewall router **210** to the specified destination.

In addition, the temporary entries to the dynamic ACLs **424, 426, 428, 430** may specify various permissible communication protocols that client **306** may use. Thus, client **306** may initiate network traffic using any communication protocol specified in its associated temporary ACL entries, as long as an idle timeout event has not occurred. The client **306** is not restricted to TCP/IP protocol that was used to send the initial HTTP request. Examples of other communication protocols that client **306** may use include telnet, Internet Control Message Protocol ("ICMP"), and File Transfer Protocol ("FTP").

Hardware Overview

FIG. **1** is a block diagram that illustrates a computer system **100** upon which an embodiment of the invention may be implemented. In one embodiment, computer system **100** is a firewall device, such as a router.

Computer system **100** includes a bus **102** or other communication mechanism for communicating information, and a processor **104** coupled with bus **102** for processing information. Computer system **100** also includes a main memory **106**, such as a random access memory (RAM) or other dynamic storage device, coupled to bus **102** for storing information and instructions to be executed by processor **104**. Main memory **106** also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor **104**. Computer system **100** further includes a read only memory (ROM) **108** or other static storage device coupled to bus **102** for storing static information and instructions for processor **104**. A storage device **110**, such as non-volatile random-access memory (NVRAM), is provided and coupled to bus **102** for storing information and instructions.

Computer system **100** may be coupled via communication interface **117** to a terminal **112**, such as a cathode ray tube (CRT) dumb terminal or workstation, for receiving command-line instructions from and displaying information to a computer user. Terminal **112** includes an input device such as a keyboard, and may include a cursor control such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor **104**.

Computer system **100** has a switching system **116** which provides a plurality of links or interfaces to a network. Switching system **116** provides a way to connect an incoming network link **114** to an outgoing network link **118**. There may be many links **114, 118**.

The invention is related to the use of computer system **100** for regulating packet traffic in an integrated services network. According to one embodiment of the invention, regulating packet traffic in an integrated services network is provided by computer system **100** in response to processor **104** executing one or more sequences of one or more instructions contained in main memory **106**. Such instructions may be read into main memory **106** from another computer-readable medium, such as storage device **110**. Execution of the sequences of instructions contained in main memory **106** causes processor **104** to perform the process steps described herein. In alternative embodiments, hardwired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor **104** for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, NVRAM, such as storage device **110**, or magnetic or optical disks. Volatile media includes dynamic memory, such as main memory **106**. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus **102**. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor **104** for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system **100** can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus **102**. Bus **102** carries the data to main memory **106**, from which processor **104** retrieves and executes the instructions. The instructions received by main memory **106** may optionally be stored on storage device **110** either before or after execution by processor **104**.

Computer system **100** also includes a communication interface **117** coupled to bus **102**. Communication interface **117** provides a two-way data communication coupling to a network link **120** that is connected to a local network **122**. For example, communication interface **117** may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface **117** may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface **117** sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link **120** typically provides data communication through one or more networks to other data devices. For example, network link **120** may provide a connection through local network **122** to a host computer **124** or to data equipment operated by an Internet Service Provider (ISP) **126**. ISP **126** in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" **128**. Local network **122** and Internet **128** both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link **120** and through communication interface **117**, which carry the digital data to and from computer system **100**, are exemplary forms of carrier waves transporting the information.

Computer system **100** can send messages and receive data, including program code, through the network(s), network link **120** and communication interface **117**. In the Internet example, a server **130** might transmit a requested code for an application program through Internet **128**, ISP **126**, local network **122** and communication interface **117**. In accordance with the invention, one such downloaded application provides for regulating packet traffic in an integrated services network as described herein.

The received code may be executed by processor **104** as it is received, and/or stored in storage device **110**, or other non-volatile storage for later execution. In this manner, computer system **100** may obtain application code in the form of a carrier wave.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

What is claimed is:

1. A computer-readable medium carrying one or more sequences of one or more instructions for controlling access of a client to a network resource using a network firewall routing device, the one or more sequences of one or more instructions including instructions which, when executed by one or more processors, cause the one or more processors to perform the steps of:

creating and storing client authorization information at the network firewall routing device that is logically interposed between the client and the network resource, wherein the client authorization information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client has with respect to the network resource;

receiving a request from the client to communicate with the network resource;

determining, at the network firewall routing device, whether the client is authorized to communicate with the network resource based on the authorization information; and

reconfiguring the network firewall routing device to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information.

2. A computer-readable medium as recited in claim **1**, wherein creating and storing client authorization information comprises the steps of creating and storing in a cache in the network routing device a set of authorization information for each client that communicates with the network routing device.

3. A computer-readable medium as recited in claim **1**, wherein creating and storing client authorization information comprises the steps of creating and storing in the network routing device an authentication cache for each client that communicates with the network routing device.

4. A computer-readable medium as recited in claim **1**, wherein creating and storing client authorization information comprises the steps of creating and storing in the network routing device a plurality of authentication caches, each authentication cache uniquely associated with one of a plurality of clients that communicate with the network routing device, each authentication cache comprising information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource.

5. A computer-readable medium as recited in claim **1**, wherein determining whether the client is authorized to communicate with the network resource comprises the step of determining whether information in the request identifying the client matches information in a filtering mechanism of the network routing device and the authorization information stored in the network routing device.

6. A computer-readable medium as recited in claim **1**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address of the client in a data packet of the request matches information in a filtering mechanism of the network routing device; and

if so, determining whether the source IP address matches the authorization information stored in the network routing device.

7. A computer-readable medium as recited in claim **1**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address of the client in a data packet of the request matches information in an a filtering mechanism of the network routing device;

if a match is found using the filtering mechanism, determining whether the source IP address matches the authorization information stored in the network routing device; and

when the source IP address fails to match the authorization information stored in the network routing device, determining if user identifying information received from the client matches a profile associated with the user that is stored in an authentication server that is coupled to the network routing device.

8. A computer-readable medium as recited in claim **1**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether client identifying information in the request matches information in a filtering mechanism of the network routing device;

if a match is found using the filtering mechanism, determining whether the client identifying information matches the authorization information stored in the network routing device; and

only when the client identifying information fails to match the authorization information stored in the network routing device, then:

creating and storing new authorization information in the network device that is uniquely associated with the client;

requesting login information from the client;

authenticating the login information by communicating with an authentication server that is coupled to the network routing device; and

updating the new authorization information based on information received from the authentication server.

9. A computer-readable medium as recited in claim **8**, wherein:

requesting login information from the client comprises sending a Hypertext Markup Language login form from the network routing device to the client to solicit a username and a user password; and

authenticating the login information by communicating with an authentication server that is coupled to the network routing device comprises determining, from a profile associated with a user of the client stored in the authentication server, whether the username and password are valid.

10. A computer-readable medium as recited in claim **1**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address in the request matches information in a filtering mechanism of the network routing device;

determining whether the source IP address matches the authorization information stored in the network routing device using an authentication cache in the network routing device; and

only when the source IP address fails to match the authorization information stored in the network routing device, then:

creating and storing a new entry in the authentication cache that is uniquely associated with the client;

requesting login information from the client;

authenticating the login information by communicating with an authentication server that is coupled to the network routing device; and

updating the new entry in the authentication cache based on information received from the authentication server.

11. A computer-readable medium as recited in claim **1**, wherein reconfiguring the network routing device comprises the steps of creating and storing one or more commands to the network routing device which, when executed by the network routing device, result in modifying one or more routing interfaces of the network routing device to permit communications between the client and the network resource.

12. A computer system for controlling access of a client to a network resource using a network firewall routing device, comprising:

one or more processor;

a storage medium carrying one or more sequences of one or more instructions including instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of:

creating and storing client authorization information at the network firewall routing device that is logically interposed between the client and the network resource, wherein the client authorization information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client has with respect to the network resource;

receiving a request from the client to communicate with the network resource;

determining, at the network firewall routing device, whether the client is authorized to communicate with the network resource based on the authorization information;

reconfiguring the network firewall routing device to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information;

wherein creating and storing client authorization information comprises the steps of creating and storing in a cache in the network routing device a set of authorization information for each client that communicates with the network routing device.

13. A computer system as recited in claim **12**, wherein creating and storing client authorization information comprises the steps of creating and storing in the network routing device a plurality of authentication caches, each authentication cache uniquely associated with one of a plurality of clients that communicate with the network routing device, each authentication cache comprising information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource.

14. A computer system as recited in claim **12**, wherein determining whether the client is authorized to communicate with the network resource comprises the step of determining whether information in the request identifying the client matches information in a filtering mechanism of the network routing device and the authorization information stored in the network routing device.

15. A computer system as recited in claim **12**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address of the client in a data packet of the request matches information in a filtering mechanism of the network routing device; and

if so, determining whether the source IP address matches the authorization information stored in the network routing device.

16. A computer system as recited in claim **12**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address of the client in a data packet of the request matches information in an a filtering mechanism of the network routing device;

if a match is found using the filtering mechanism, determining whether the source IP address matches the authorization information stored in the network routing device; and

when the source IP address fails to match the authorization information stored in the network routing device, determining if user identifying information received from the client matches a profile associated with the user that is stored in an authentication server that is coupled to the network routing device.

17. A computer system as recited in claim **12**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether client identifying information in the request matches information in a filtering mechanism of the network routing device;

if a match is found using the filtering mechanism, determining whether the client identifying information matches the authorization information stored in the network routing device; and

only when the client identifying information fails to match the authorization information stored in the network routing device, then:

creating and storing new authorization information in the network device that is uniquely associated with the client;

requesting login information from the client;

authenticating the login information by communicating with an authentication server that is coupled to the network touting device; and

updating the new authorization information based on information received from the authentication server.

18. A computer system as recited in claim **17**, wherein:

requesting login information from the client comprises sending a Hypertext Markup Language login form from the network routing device to the client to solicit a username and a user password; and

authenticating the login information by communicating with an authentication server that is coupled to the network routing device comprises determining, from a profile associated with a user of the client stored in the authentication server, whether the username and password are valid.

19. A computer system as recited in claim **12**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address in the request matches information in a filtering mechanism of the network routing device;

determining whether the source IP address matches the authorization information stored in the network routing device using an authentication cache in the network routing device; and

only when the source IP address fails to match the authorization information stored in the network routing device, then:

creating and storing a new entry in the authentication cache that is uniquely associated with the client;

requesting login information from the client;

authenticating the login information by communicating with an authentication server that is coupled to the network routing device; and

updating the new entry in the authentication cache based on information received from the authentication server.

20. A computer system as recited in claim **12**, wherein reconfiguring the network routing device comprises the steps of creating and storing one or more commands to the network routing device which, when executed by the network routing device, result in modifying one or more routing interfaces of the network routing device to permit communications between the client and the network resource.

21. A data packet firewall router that is logically interposed between a client and a network resource and that controls access of the client to the network resource, comprising:

one or more processors;

a storage medium carrying one or more sequences of one or more instructions including instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of:

creating and storing client authorization information at the router, wherein the client authentication information comprises information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client has with respect to the network resource;

receiving a request from the client to communicate with the network resource;

determining, at the router, whether the client is authorized to communicate with the network resource based on the authorization information;

reconfiguring the router to permit the client to communicate with the network resource only when the client is authorized to communicate with the network resource based on the authorization information;

wherein creating and storing client authorization information comprises the steps of creating and storing in a cache in the network routing device a set of authorization information for each client that communicates with the network routing device.

22. A data packet router as recited in claim **21**, wherein creating and storing client authorization information comprises the steps of creating and storing in the network routing device a plurality of authentication caches, each authentication cache uniquely associated with one of a plurality of clients that communicate with the network routing device, each authentication cache comprising information indicating whether the client is authorized to communicate with the network resource and information indicating what access privileges the client is authorized to have with respect to the network resource.

23. A data packet router as recited in claim **21**, wherein determining whether the client is authorized to communicate with the network resource comprises the step of determining whether information in the request identifying the client matches information in a filtering mechanism of the network routing device and the authorization information stored in the network routing device.

24. A data packet router as recited in claim **21**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address of the client in a data packet of the request matches information in a filtering mechanism of the network routing device; and

if so, determining whether the source IP address matches the authorization information stored in the network routing device.

25. A data packet router as recited in claim **21**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address of the client in a data packet of the request matches information in an a filtering mechanism of the network routing device;

if a match is found using the filtering mechanism, determining whether the source IP address matches the authorization information stored in the network routing device; and

when the source IP address fails to match the authorization information stored in the network routing device, determining if user identifying information received from the client matches a profile associated with the user that is stored in an authentication server that is coupled to the network routing device.

26. A data packet router as recited in claim **21**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether client identifying information in the request matches information in a filtering mechanism of the network routing device;

if a match is found using the filtering mechanism, determining whether the client identifying information matches the authorization information stored in the network routing device; and

only when the client identifying information fails to match the authorization information stored in the network routing device, then:

creating and storing new authorization information in the network device that is uniquely associated with the client;

requesting login information from the client;

authenticating the login information by communicating with an authentication server that is coupled to the network routing device; and

updating the new authorization information based on information received from the authentication server.

27. A data packet router as recited in claim **26**, wherein:

requesting login information from the client comprises sending a Hypertext Markup Language login form from the network routing device to the client to solicit a username and a user password; and

authenticating the login information by communicating with an authentication server that is coupled to the network routing device comprises determining, from a profile associated with a user of the client stored in the authentication server, whether the username and password are valid.

28. A data packet router as recited in claim **21**, wherein determining whether the client is authorized to communicate with the network resource comprises the steps of:

determining whether a source IP address in the request matches information in a filtering mechanism of the network routing device;

determining whether the source IP address matches the authorization information stored in the network routing device using an authentication cache in the network routing device; and

only when the source IP address fails to match the authorization information stored in the network routing device, then:

creating and storing a new entry in the authentication cache that is uniquely associated with the client;

requesting login information from the client;

authenticating the login information by communicating with an authentication server that is coupled to the network routing device; and

updating the new entry in the authentication cache based on information received from the authentication server.

29. A data packet router as recited in claim **21**, wherein reconfiguring the network routing device comprises the steps of creating and storing one or more commands to the network routing device which, when executed by the network routing device, result in modifying one or more routing interfaces of the network routing device to permit communications between the client and the network resource.

* * * * *

US006549773B1

(12) **United States Patent**
Linden et al.

(10) **Patent No.: US 6,549,773 B1**
(45) **Date of Patent: Apr. 15, 2003**

(54) **METHOD FOR UTILIZING LOCAL RESOURCES IN A COMMUNICATION SYSTEM**

(75) Inventors: **Mikael Linden**, Tampere (FI); **Teemu Kurki**, Lempäälä (FI)

(73) Assignee: **Nokia Mobile Phones Limited**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/399,579**

(22) Filed: **Sep. 20, 1999**

(30) **Foreign Application Priority Data**

Sep. 21, 1998 (FI) 982031

(51) **Int. Cl.**^{7} **H04Q 7/20**

(52) **U.S. Cl.** **455/426**; 455/412; 455/517; 455/558; 370/328; 370/338; 370/401; 709/230; 709/227; 709/228

(58) **Field of Search** 455/558, 412, 455/517, 426; 370/328, 338, 401; 709/230, 227, 228

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,868,846	A		9/1989	Kemppi	379/144
5,266,782	A		11/1993	Alanara et al.	235/380
5,315,635	A		5/1994	Mukari	379/58
5,353,328	A		10/1994	Jokimies	379/58
5,448,622	A		9/1995	Huttunen	379/58
5,487,084	A		1/1996	Lindholm	375/215
5,600,708	A		2/1997	Meche et al.	379/59
5,956,633	A		9/1999	Janhila	455/410
6,011,976	A	*	1/2000	Michaels et al.	455/466
6,253,326	B1	*	6/2001	Lincke et al.	713/201

FOREIGN PATENT DOCUMENTS

WO	WO 94/30023	12/1994
WO	WO 97/36437	10/1997
WO	WO 98/27767	6/1998

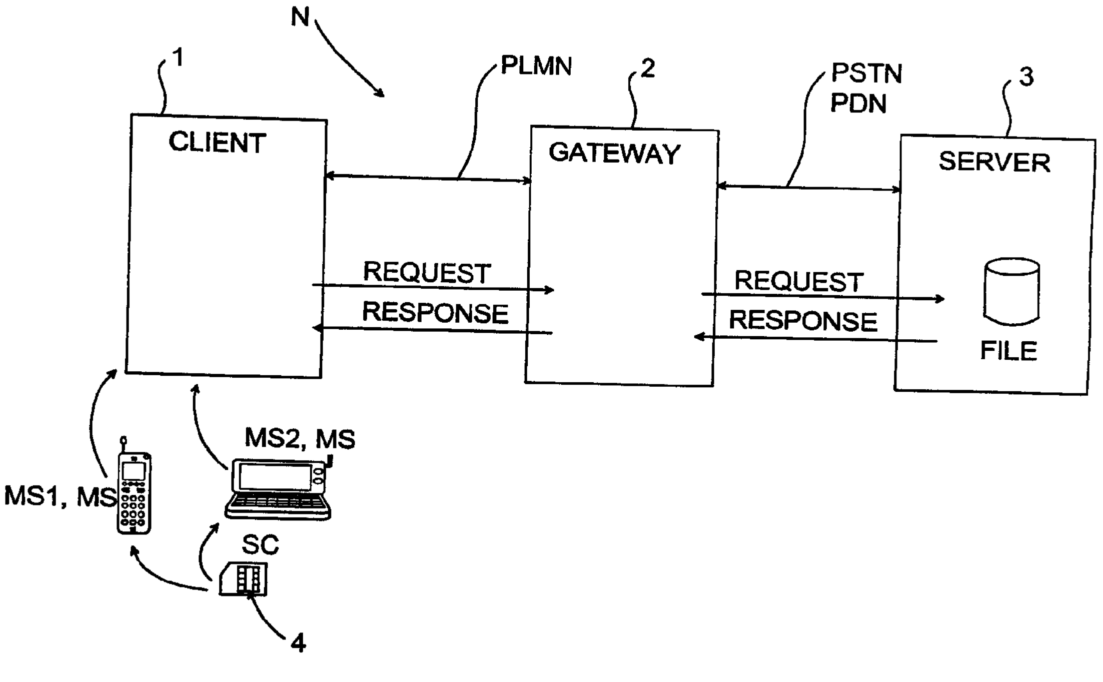
* cited by examiner

Primary Examiner—William Trost
Assistant Examiner—Danh Le
(74) *Attorney, Agent, or Firm*—Perman & Green, LLP

(57) **ABSTRACT**

The invention relates to a method in data transmission between a first mobile station (MS, MS1, MS2), a second mobile station (MS, MS1, MS2), and advantageously also a server (**3**, SERVER), wherein the first mobile station (MS, MS1, MS2) and the second mobile station (MS, MS1, MS2) comprise protocol means (**100–106**) for generating and directing a request (REQUEST) which contains at least address information (URI, URL, URN) for identifying the destination of the request (REQUEST), and wherein at least the first mobile station (MS, MS1, MS2) comprises a first local resource (SC, **4**), such as a smart card, connected to the same. In the information, the address information (URI, URL, URN) of the request (REQUEST) is established to identify said first local resource (SC, **4**), and the request (REQUEST) is generated and directed at least partly with the help of said protocol means (**100–106**).

17 Claims, 7 Drawing Sheets



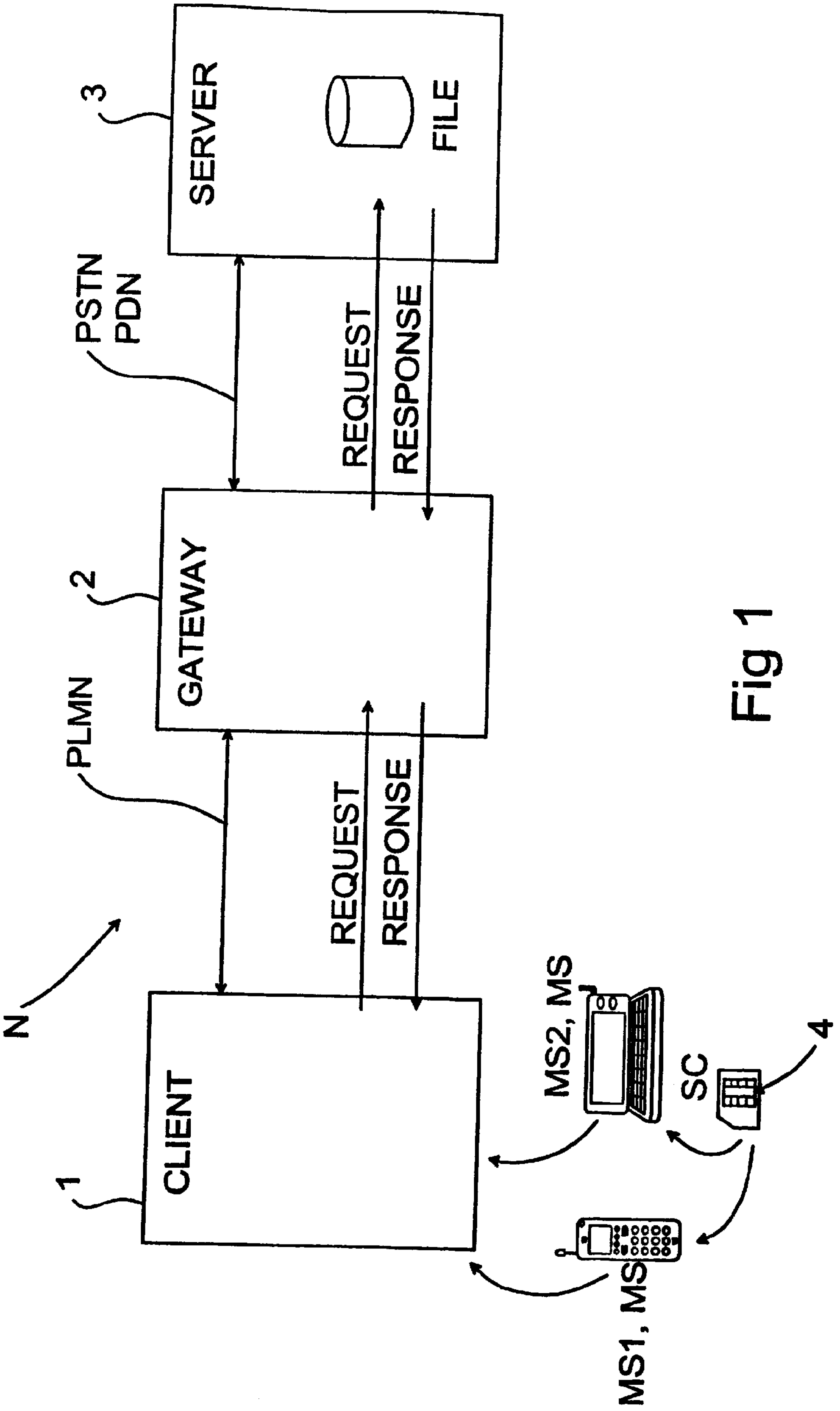

SERVER
3
FILE
PSTN
PDN
REQUEST
RESPONSE
GATEWAY
2
PLMN
REQUEST
RESPONSE
N
CLIENT
1
MS2, MS
SC
4
MS1, MS

Fig 1

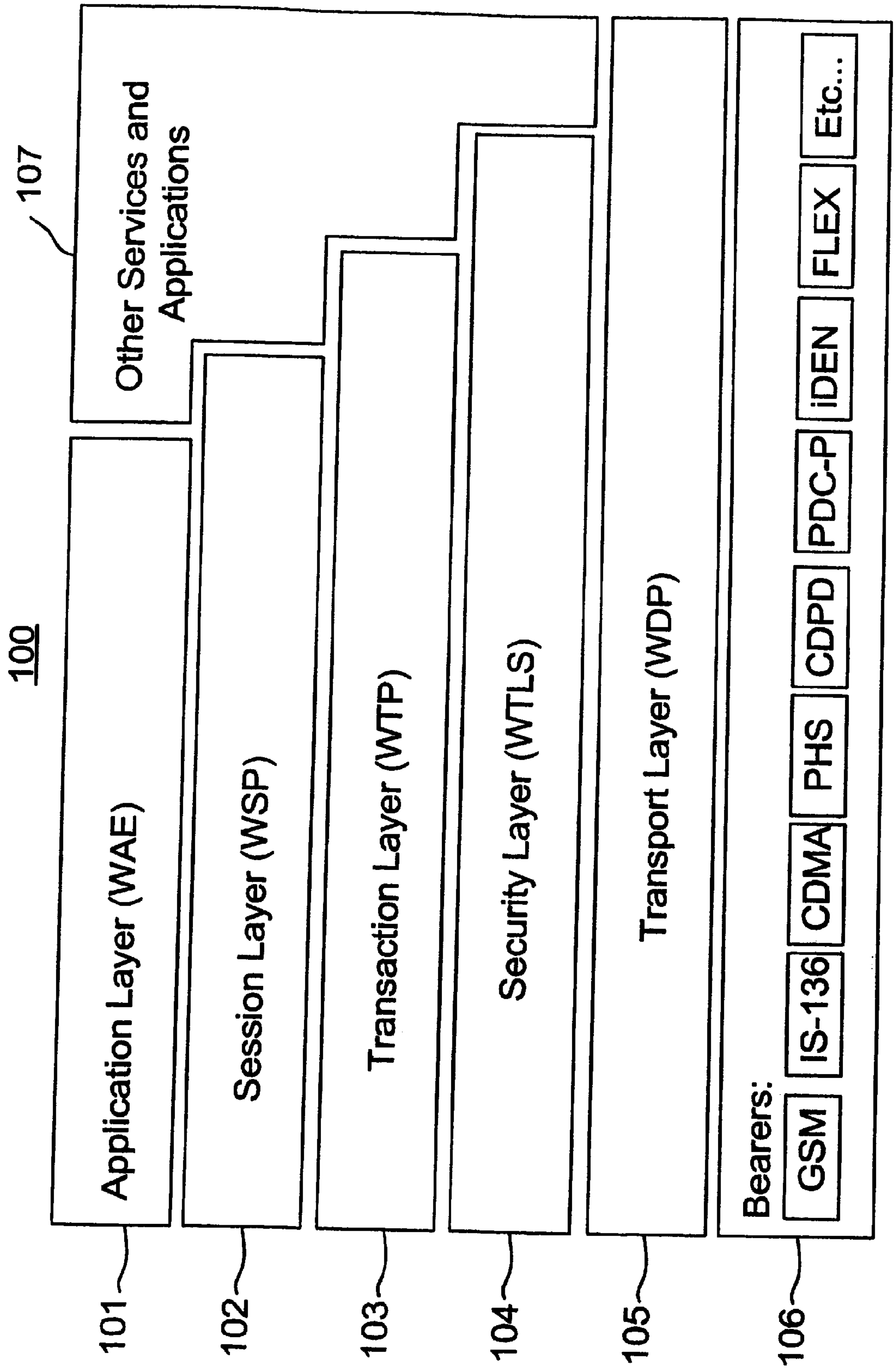

100
107
Other Services and Applications
Application Layer (WAE)
Session Layer (WSP)
Transaction Layer (WTP)
Security Layer (WTLS)
Transport Layer (WDP)
Bearers:
GSM
IS-136
CDMA
PHS
CDPD
PDC-P
iDEN
FLEX
Etc...
101
102
103
104
105
106

Fig 2a

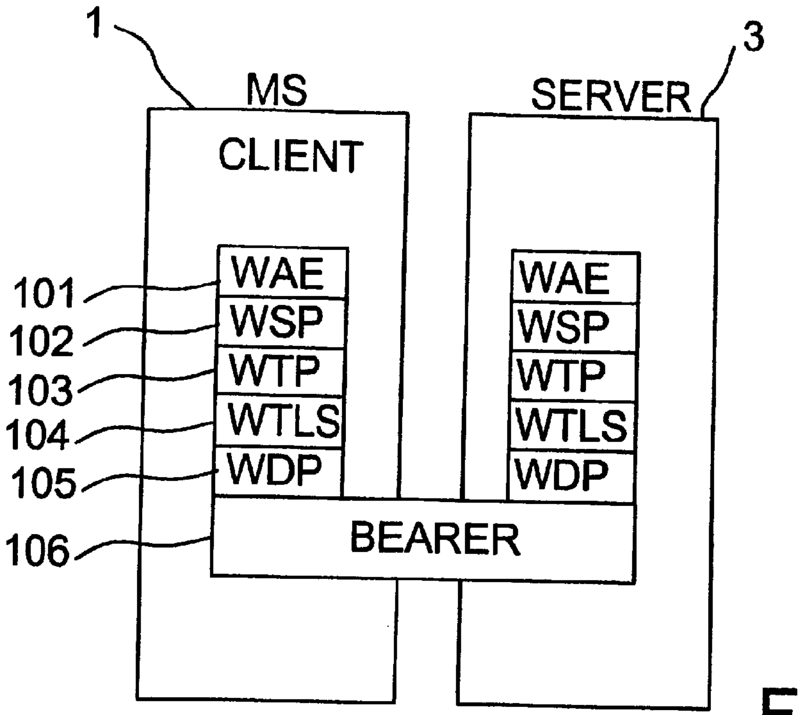

1
MS
CLIENT
3
SERVER
101
102
103
104
105
106
WAE
WSP
WTP
WTLS
WDP
BEARER
WAE
WSP
WTP
WTLS
WDP

Fig 2b

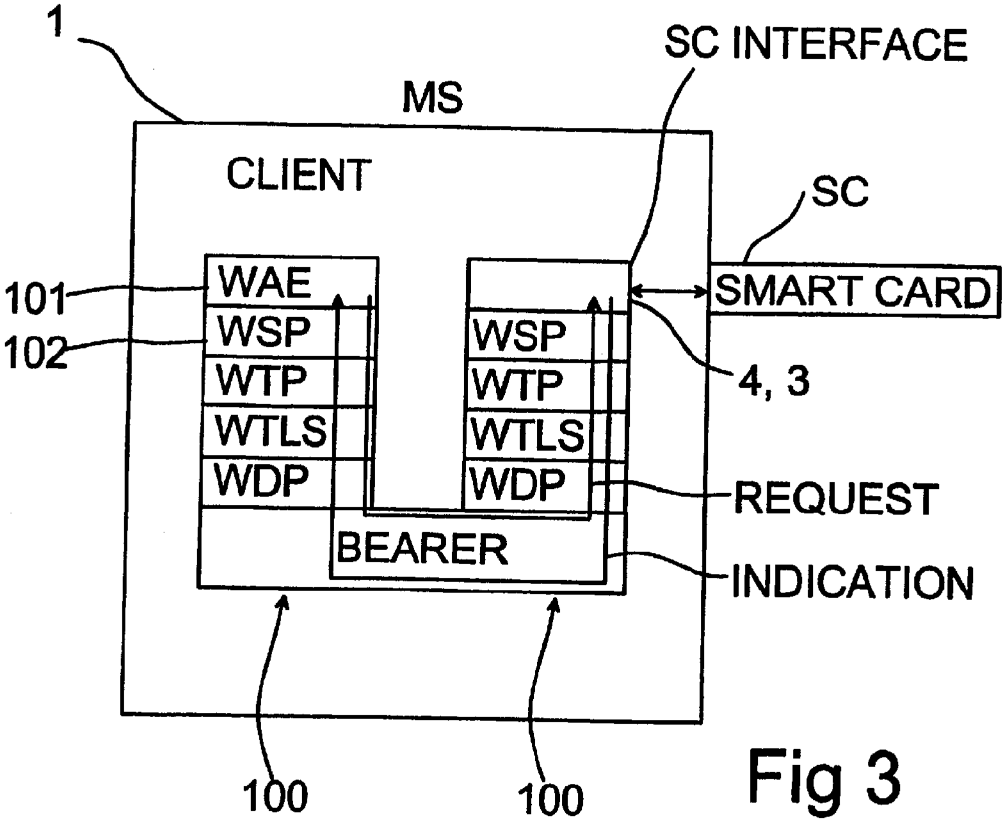

1
MS
CLIENT
SC INTERFACE
SC
SMART CARD
101
102
WAE
WSP
WTP
WTLS
WDP
BEARER
WSP
WTP
WTLS
WDP
4, 3
REQUEST
INDICATION
100
100

Fig 3

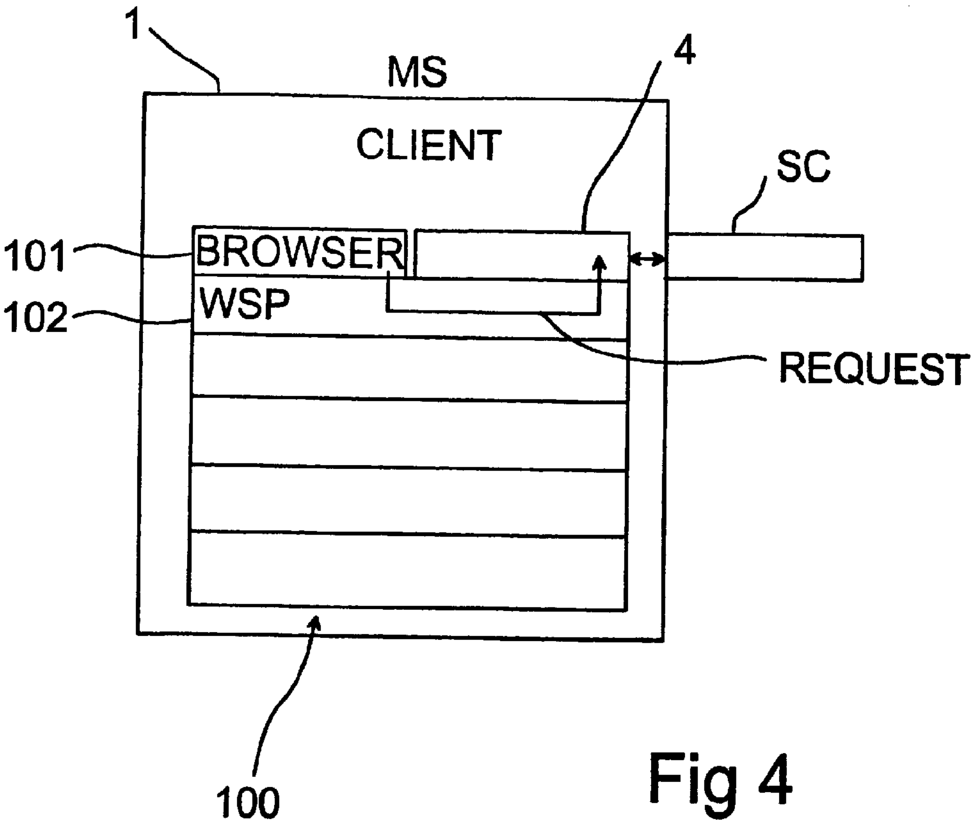

1
MS
4
CLIENT
SC
101
BROWSER
102
WSP
REQUEST
100

Fig 4

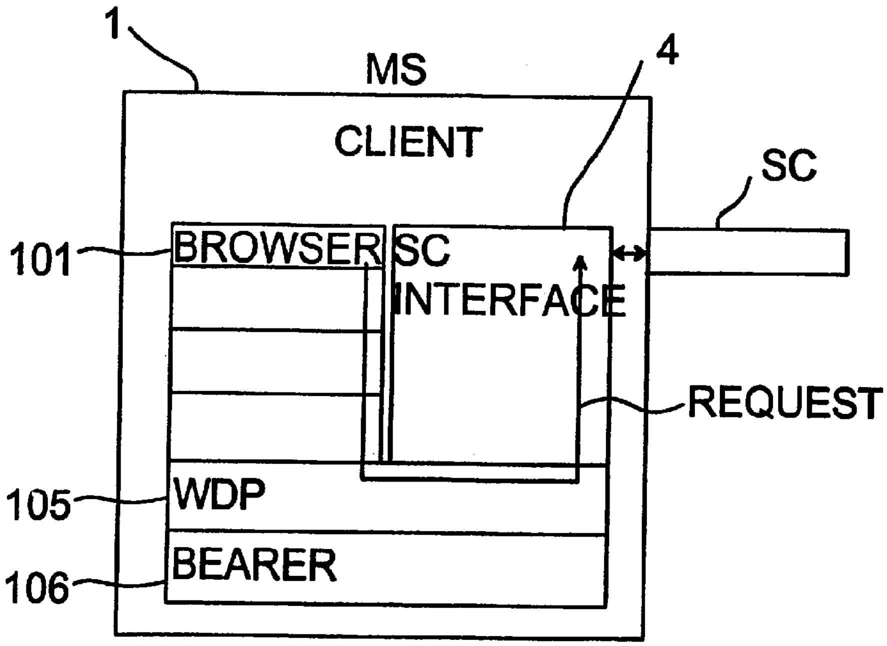

1
MS
4
CLIENT
SC
101
BROWSER
SC
INTERFACE
REQUEST
105
WDP
106
BEARER

Fig 5

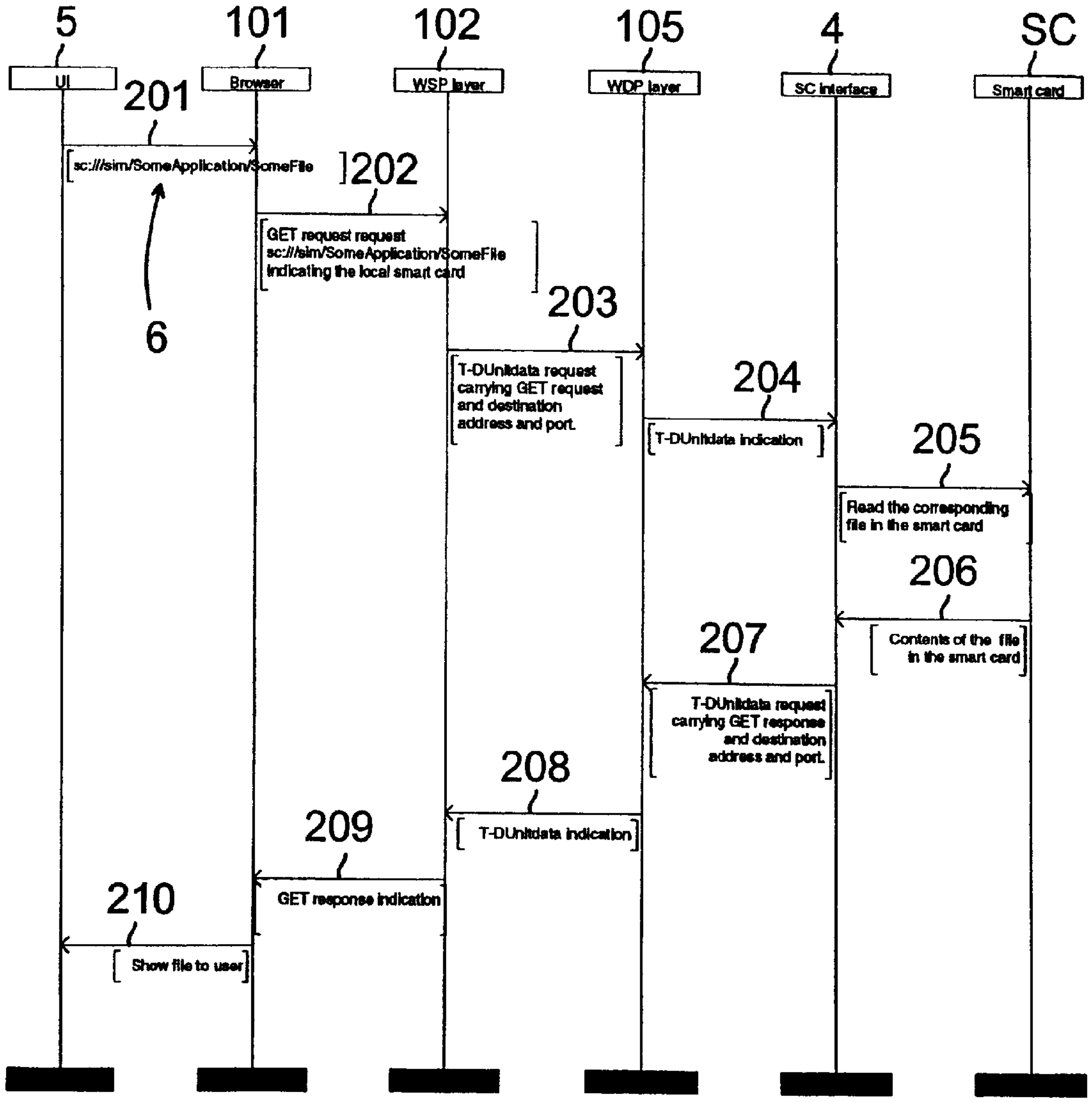

5
UI
101
Browser
102
WSP layer
105
WDP layer
4
SC interface
SC
Smart card
201
sc:///sim/SomeApplication/SomeFile
6
202
GET request request
sc:///sim/SomeApplication/SomeFile
indicating the local smart card
203
T-DUnitdata request
carrying GET request
and destination
address and port
204
T-DUnitdata indication
205
Read the corresponding
file in the smart card
206
Contents of the file
in the smart card
207
T-DUnitdata request
carrying GET response
and destination
address and port
208
T-DUnitdata indication
209
GET response indication
210
Show file to user

Fig 6

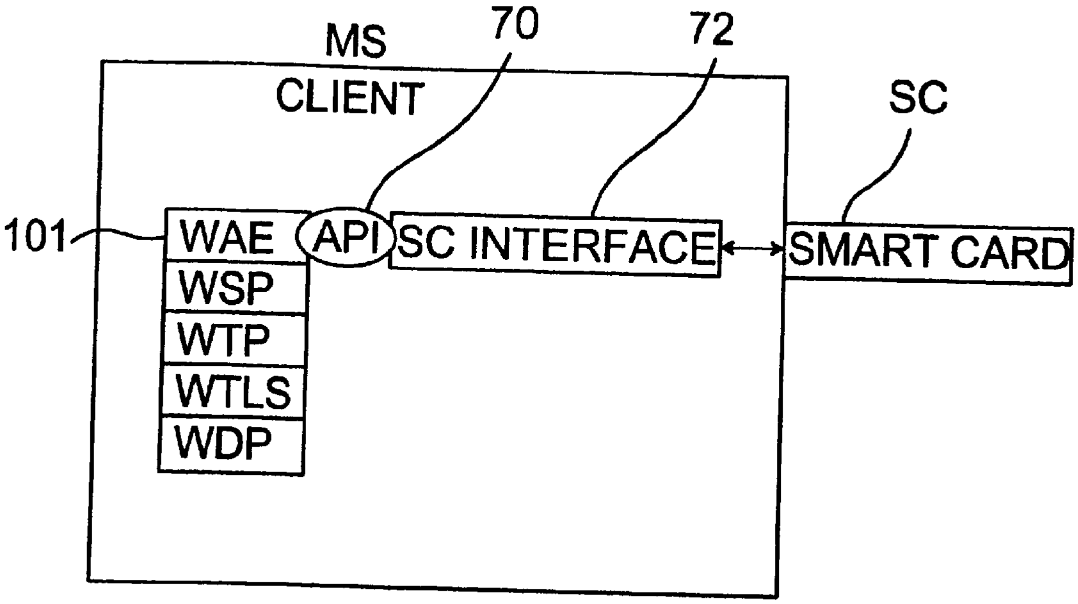

MS
70
72
CLIENT
SC
101
WAE
API
SC INTERFACE
SMART CARD
WSP
WTP
WTLS
WDP

Fig 7a

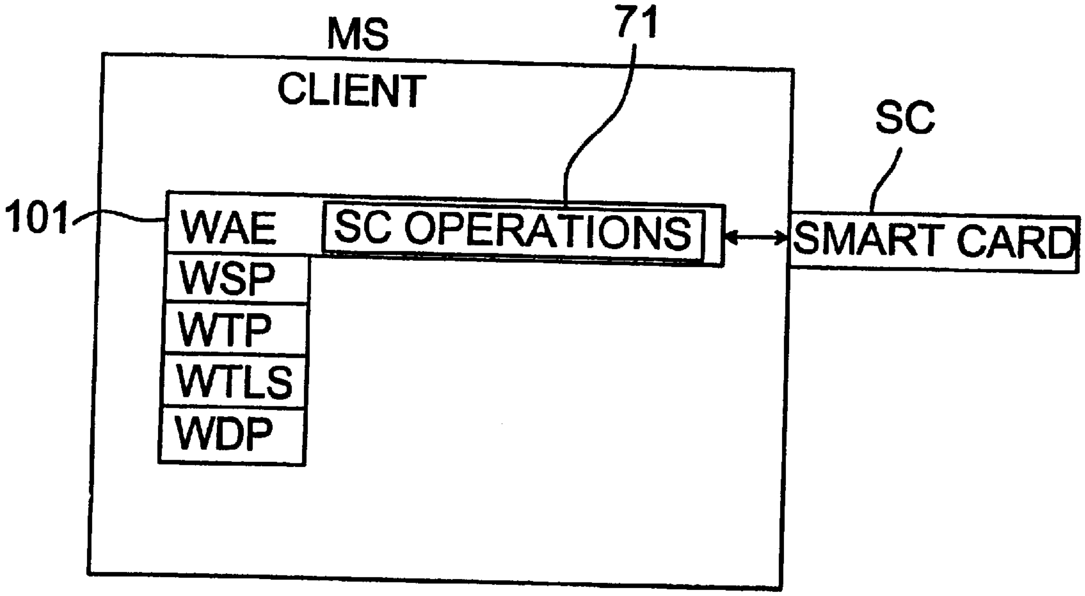

MS
71
CLIENT
SC
101
WAE
SC OPERATIONS
SMART CARD
WSP
WTP
WTLS
WDP

Fig 7b

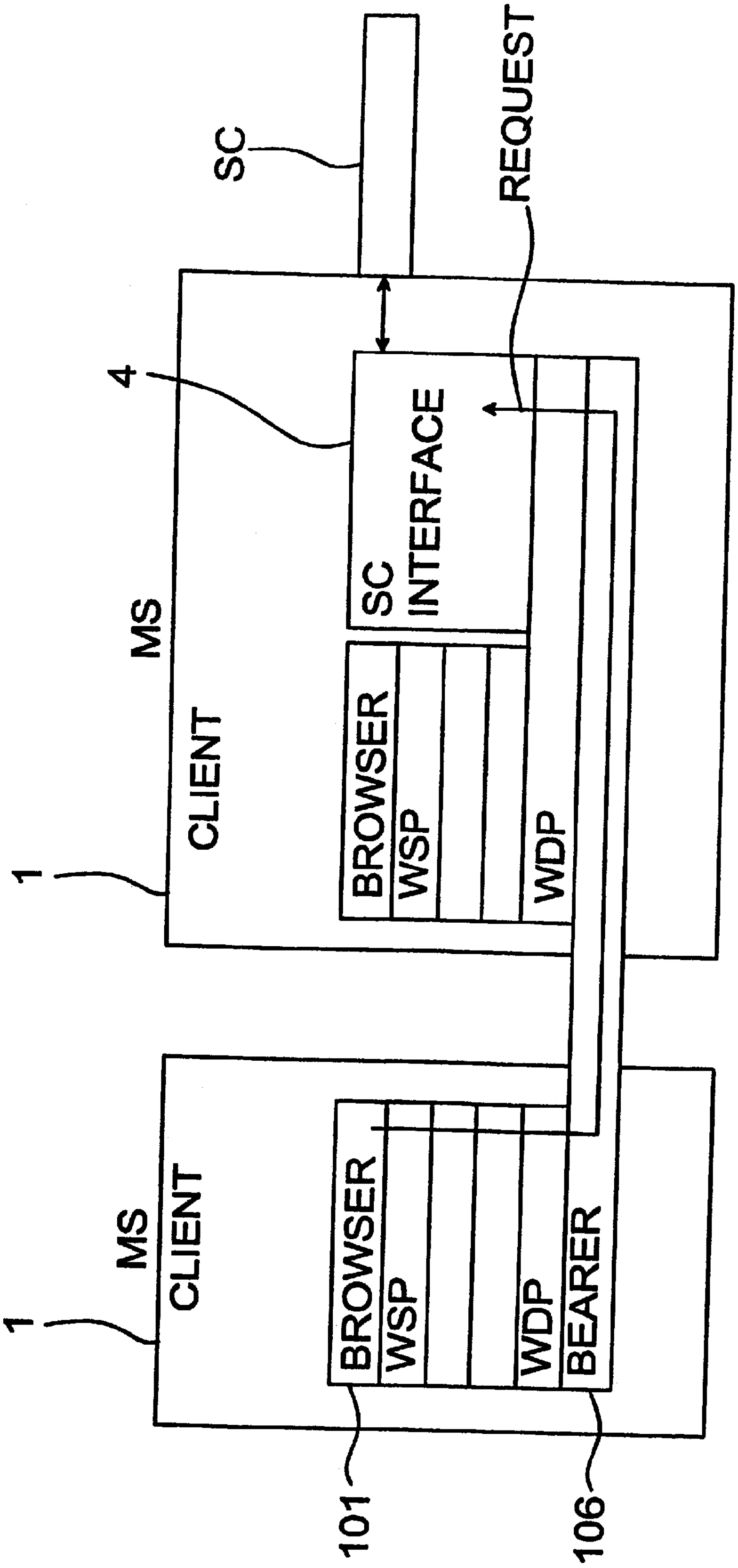

SC
REQUEST
4
MS
CLIENT
SC
INTERFACE
BROWSER
WSP
WDP
1
MS
CLIENT
BROWSER
WSP
WDP
BEARER
1
101
106

Fig 8

METHOD FOR UTILIZING LOCAL RESOURCES IN A COMMUNICATION SYSTEM

The present invention relates to a method in a communication system according to the preamble of claim **1**. The invention also relates to a communication system according to the preamble of claim **11**. Furthermore, the invention relates to a wireless communication device according to the preamble of claim **12**.

There are known wireless communication systems, such as the PLMN (Public Land Mobile Network), which is a communication network based on a cellular system. One example that can be mentioned is the GSM **900** mobile communication network according to the GSM standard (Global System for Mobile Communications). The cells of the communication network are distributed within a wide geographical area, and mobile stations (MS), such as mobile phones, which are connected to the communication network via base stations BS, move from one cell to another. These mobile phones are distinguished from each other by means of a subscriber-specific identification code, wherein communication, such as data transmission or an audio call, is possible between two mobile stations. The identification code is, for instance, an IMSI code (International Mobile Subscriber Identity). The communication network takes care of routing information via base stations and mobile services switching centers (MSC), by utilizing register data which indicate the location of the mobile station in the area of the cells of different base stations. Furthermore, the following wireless communication networks should be mentioned: GSM-1800, GSM-1900, PDC, CDMA, US-TDMA, IS-95, USDC (IS-136), iDEN (ESMR), DataTAC, and Mobitex.

In order to perform data transmission and processes connected with data transmission in communication devices, such as servers and wireless communication devices, which are connected to a communication network, a set of communication rules must be available for defining the allowed messages and the function of the participants of the data transmission at the different stages of the communication. As is well known, one such set of communication rules in data transmission is a protocol used by the devices to communicate with each other. For data transmission especially in wireless communication networks, a wireless application protocol WAP is developed, which will be used as an example in the following specification. One version of the WAP application protocol is specified in the WAP Architecture Version Apr. 30, 1998 publication (Wireless Application Protocol Architecture Specification; Wireless Application Protocol Forum Ltd, 1998), which is published in the Internet, and which includes for example a description on the architecture of the WAP application protocol. By means of the WAP application protocol, it is possible to define a series of protocols on different levels, which can be used to develop new services and devices e.g. for digital mobile communication networks based on a cellular network. For example, the WAP application protocol has already been developed for SMS services (Short Messaging Service), USSD services (Unstructured Supplementary Services Data), CSD services (Circuit Switched Data), and GPRS services (Global Packet Radio System) of the GSM network, and for the services of the IS-136 and PDC network.

The WAP application protocol is designed to describe those standard components that enable data transmission especially between mobile stations (client) and servers of the communication network (origin server). In order to gain access to servers located in the WWW network, the WAP uses a gateway which also functions as a proxy containing functions for data transmission between a WAP protocol stack and a WWW protocol stack (HTTP, TCP/IP), as well as functions for coding and decoding the content (WML, Wireless Markup Language, or HTML) of the information for data transmission. In the WAP, specified presentation formats are used to define the content of the information and the applications. The content is transferred using standardized data transmission protocols. A so-called browser or a microbrowser is used in the wireless communication device to control a user interface (UI).

The application layer in the architecture of the aforementioned WAP application protocol applies a defined architecture of a wireless application environment WAE. The purpose of the WAE application environment is to provide operators and service providers with an open environment, by means of which it is possible to create a large group of services and applications on top of different wireless communication methods functioning as a platform. The different WAE applications of communication devices follow a procedure used in the Internet World Wide Web (WWW) network, in which different applications and information are presented by means of standardized presentation formats and browsed for example with known WWW browsers. Consequently, in order to use different resources of communication devices, the servers and the information of the Internet network are labelled with a URI address (Uniform Resource Identifier) which is independent of the location, and the presentation format of the information is supported by the browser used, and is, for example, HyperText Markup Language (HTML) or JavaScript. On the other hand, the WAE application environment especially takes into account the requirements of the wireless communication devices and wireless communication networks. At present, according to prior art, the WAE applications (user agents), such as browsers, only support the WSP/B protocol. For example browsers communicate with a gateway server via a WSP layer (Wireless Session Protocol) of the WAP protocol stack, which will be described later. The gateway, in turn, provides functions for converting the data transmission protocol so that access to the resources of a WWW server using the HTTP protocol would be possible. According to prior art, the WAP protocol stack is described in more detail in the aforementioned publication, and the WAE application environment is described in more detail e.g. in the WAP WAE Version Apr. 30, 1998 publication, published in the Internet (Wireless Application Protocol Wireless Application Environment Overview; Wireless Application Protocol Forum Ltd, 1998).

The URI addresses are used to locate resources by providing the location of the resource with an abstract identification. When the resourse is located, the system can subject the resource to different procedures which depend on the application and on the purpose of pursuing access to the resource. As is well known, several different data transmission protocols are used in this context, of which for example the HTTP (HyperText Transport Protocol), FTP (File Transfer Protocol), MAILTO (Electronic Mail Address), and GOPHER (The Gopher Protocol) can be mentioned.

As is well known, the URL address used by the HTTP is utilized to indicate resources which are available in the Internet network, for example in its servers, by using an HTTP data transmission protocol, and it has the format:

http://<host>:<port>/<path>,

in which the data transmission protocol used can be deduced from the "http" part, "<host>" represents the domain name or the IP address (Internet Protocol) of the server in the communication network, "<port>" represents the number of the port, and it can also be left out, because data transmission protocols use a default port. Furthermore, "<path>" describes the resource in question in more detail and functions as a selector in the HTTP. The prefix "//" illustrates that the address follows the data transmission protocols used in the Internet network. A more precise indication of the resources by means of the "<path>" part varies in different data transmission protocols, and in addition to that, it is possible to provide a "<user>" part between the "//" and "<host>" parts to indicate the user, as in the FTP, and a "<password>" part to indicate a password. The resource can also be identified by means of a URN name (Uniform Resource Naming), wherein it is possible to use only a name instead of a URL address, and the URN name is modified to a URL address when necessary. The URI address and the protocol (access algorithm) to be used, form a URL address (Uniform Resource Locator) to identify the resource.

As is well known, resources refer to a area (such as a directory or a file), a program (such as an application), or a peripheral device (such as a printer) allocated in the server of the data transmission network for collective use. In the prior art, however, problems often occur which relate especially to the use of these local resources. These local resources can include for example content and files, as well as applications contained in the same communication device, and peripheral and auxiliary devices connected to the same. Local resources can also be located in a peripheral device, such as a SIM card (Subscriber Identity Module) or a smart card, connected to a wireless terminal, such as a mobile phone. At present, for example the WAP application protocol contains no specification for methods for utilizing these local resources coupled to a wireless communication device.

In connection with WWW communication networks, there are such known agents as for example the WWW browsers Netscape Navigator and Microsoft Internet Explorer. A function known from the Navigator browser is the viewing of local files which are stored in the local hard disc of a data transmission device, typically a computer. The browser opens a file stored on the hard disc, reads its content and presents it in an intelligible format to the user. This browsing of files is implemented as an extension to the browser. Consequently, all necessary command primitives and references to the services (i.e. system functions) provided by the operating system software controlling the actual operation of the device, are built in the program code of the browser in order to implement the browsing of files. This can be implemented for example by means of a so-called standardized API interface, i.e. an application programming interface.

In the WAE application environment, the above-described facts would entail for example that an API interface was arranged for each WAE application of a wireless communication device for coupling to a smart card. Correspondingly, the interface of the smart card would be responsible for transmitting low-level commands (command APDU) to the smart card. The responses given by the card are then transmitted back to the application via interfaces. Known methods to attend to the coupling are the Open Card Framwork specification and the PC/SC specification. However, the specifications differ from each other, and thus each WAE application should take the differences and various API interfaces into account.

Another possibility is to arrange the WAE application in such a way that it would be in direct communication with a resource, such as a smart card, which is physically coupled to the mobile station. This would mean that the WAE application would be responsible for transmitting command primitives (command APDU), the functions being programmed in the WAE application. This would create a significant problem, because it would be necessary to provide each WAE application with the necessary command primitives for each different local resource. This would also have the result that for new resources, the WAE applications would have to be supplemented with the necessary command primitives so that these resources could be implemented. That would create a significant need to update the WAE applications, a limited possibility to use the new resources, and a barrier to the implementation and development of new resources.

The purpose of the present invention is to provide a method that enables applications utilizing the used application protocol to gain efficiently access to local resources in a straightforward and efficient manner. A particular purpose of the invention is to introduce a new method to enable applications applying the WAP application protocol to gain access to local resources, which include for example a SIM card of a mobile station, or an attachable smart card.

The central principle of the invention is to utilize the protocol stack used in order to utilize local resources. Another central principle of the invention is to apply an indication corresponding e.g. to the URL address when referring to local resources.

The method in the communication system according to the invention is characterized in what will be presented in the characterizing part of the appended claim 1. The communication system according to the invention is characterized in what will be presented in the characterizing part of the appended claim 11. The wireless communication device according to the invention is characterized in what will be presented in the characterizing part of the appended claim 12.

A considerable advantage of the invention is, that it is possible to avoid including a standardized API interface or necessary command primitives in the application, and nevertheless, local resources can be utilized flexibly. Thus, for example the complexity of WAP applications is reduced considerably, and the implementation of applications in the mobile station is facilitated as well. The changes required to implement the invention in WAE applications are small, because the mechanisms already available in the WAE application (GET and POST methods, etc) are utilized. In the mobile station, the changes required by the invention can be advantageously implemented with changes in the application software which controls the functions of the mobile station.

The principle of the WAP application protocol is that a mobile station functioning as a client contacts a server which is located in the communication network. The advantage of the invention is that this principle can still be applied, wherein to gain access to the local resources, the application makes a request to the lower protocol layer and also receives a response which the server, in this case a local resource, has generated and transmitted to the client. A further advantage of the invention is that the principle can also easily be expanded for example in such a way that the local resource can receive requests from another mobile station or server too, not only from the one containing the smart card. In that case, the interface of the smart card functions like a server.

The local resource can also transmit requests to other mobile stations and servers, and, furthermore, also to the mobile station to which it is attached. The advantage therein is that the smart card, such as a SIM card, can transmit a request to the server, for example to update applications functioning in a SIM card from the server of the operator (download).

The principle of the invention can also be implemented in a situation where the smart card is communicating with another smart card. An example that can be mentioned is the transfer and relay of payments from one smart card to another. Thus, one of the smart cards can be connected to a server functioning in the network, and a protocol stack according to the invention is implemented in the server. By means of the "<host>" part of the URL address, it is possible to indicate said server. The user attaches the smart card into the mobile station, after which an application is activated for setting up a connection to the server, the application being loaded from the server and implemented for example by means of a WMLScript command language.

A particular advantage of the invention e.g. in connection with the WAP application protocol is that it is possible to efficiently utilize functions connected with the HTTP data transmission protocol of the WSP/B protocol already known as such. These include, for example, GET, PUT, and POST requests. Consequently, the header fields of the HTTP protocol can also be utilized in the data transmission, as well as the headers of the HTTP protocol for authentication. Correspondingly, it is possible to utilize efficiently the methods of the WWW communication network for authorization or data transmission.

A further advantage of the invention is that it can be used to conceal the structure and implementation of the local resource from the user of said resource. When several implementations of the local resource, for example a memory module, a smart card or both of them, are coupled to the mobile phone, their lower level interfaces typically are very different. By means of the invention it is possible to use them in the same way, because both resources appear to the user as servers with which communication is effected by means of the WSP/B protocol and via the interface (Local Resource Interface). Thus, the user does not have to understand the real structure of the resource, and the low level interface through which the communication takes place.

In the following, the present invention will be described in more detail with reference to the appended figures, in which

FIG. **1** is a reduced diagram showing a communication system applying the invention,

FIG. **2***a* shows a protocol stack of an application protocol applying the invention,

FIG. **2***b* is a reduced diagram showing the application and logical structure of a protocol stack in the application protocol presented in FIG. **2***a* in implementing the client/server hierarchy,

FIG. **3** is a reduced diagram showing the application and logical structure of the client/server hierarchy presented in FIG. **2***b* in the same wireless communication device,

FIG. **4** is a diagram showing an implementation according to a first preferred embodiment of the invention,

FIG. **5** is a diagram showing an implementation according to a second preferred embodiment of the invention, and

FIG. **6** is a sequential diagram showing how a request and a response are generated and directed according to the invention in the embodiment according to FIG. **5**,

FIGS. **7***a*–**7***b* illustrate alternative implementations for utilizing a local resource in connection of the protocol stack presented in FIG. **2***a*, and

FIG. **8** is a diagram showing an implementation according to a third preferred embodiment of the invention.

FIG. **1** is a reduced diagram showing a communication system known as such in which it is possible to apply the invention. Communication devices, i.e. wireless communication devices **MS1** and **MS2**, advantageously mobile stations (MS) function as clients **1** and are connected to a gateway **2**, which is advantageously a server and which adapts the different data transmission protocols used to each other. The clients **1** utilize advantageously a public land mobile network (PLMN), such as the GSM network and the GSM GPRS network, in order to implement wireless data transmission. The base station subsystem (BSS) of the mobile communication network (PLMN) is known as such and comprises base transceiver stations (BTS) and base station controllers (BSC). The mobile station **MS1**, **MS2** communicates with a base transceiver station via a radio channel, and the base transceiver station communicates further with a base station controller. The base station controller, in turn, communicates with a mobile services switching center (MSC). Mobile services switching centers can, in turn, communicate with each other and with servers in a public switched telephone network (PSTN). The base station controller can also communicate with a public packet data network (PDN). The aforementioned origin or content server **3** can be located either in the PSTN network or in the PDN network, wherein the gateway server **2** uses these networks in data transmission. The server **2** can communicate either with the base station subsystem or with the mobile services switching center in the PLMN network. Thus, the server **2** can be located either in the PLMN network itself or in the PSTN network, and it is obvious that the content server **3** and the gateway server **2** can be physically located in the same communication device. Furthermore, it is obvious that to implement the gateway **2**, several separate servers can be used.

The invention can also be applied in a communication system in which the communication between different data transmission devices, such as servers **2** and clients **1**, takes place by means of short distance IR data transmission (infrared), LPRF data transmission (Low Power Radio Frequency), SDRF data transmission (Short Distance Radio Frequency), or inductive data transmission, wherein data transmission distances in the range of a single communication network are typically shorter than in the mobile communication network.

Smart cards are typically small cards manufactured in the size of a credit card, which, laminated in plastic, contains a micro controller as well as electronic circuits and memory circuits required for the function of the micro controller. Furthermore, the surface of the card typically contains electrical contacts, via which it is possible to transmit operating voltages to the card and to transfer control and data signals between the card and a reading/writing device for the card. There are also known methods in which signals and operating voltages of the card are transmitted wirelessly, for example as high frequency electromagnetic signals, between the card and the reading/writing device for the card. Smart cards are used for example as charge cards in various applications, for example in public telephones, as change cards, as a means of payment in public transport, etc. Asmart card used in mobile phones is a so-called SIM card, which in modern mobile phones is typically a small-sized mini-card inserted in the telephone. The function of the card is, for instance, to store subscriber data and identifications (PIN, Personal Identification Number), and thus the card determines the subscriber number of the telephone. The SIM card can also contain a stored list of telephone numbers or a group of short messages, as well as various data and set values related to the communication network used. This data is utilized increasingly by different applications at the same time when the quantity of data to be stored in the smart card is increased and diversified.

In the following specification, the smart card is used as an example of a local resource in connection with which it is possible to apply the invention. It is, of course, obvious that within the scope of the claims, the invention can also be applied to utilize another local resource, in order to avoid the above described problems and to gain advantages.

FIG. **1** also presents a smart card SC, known as such, which is a mini-card, i.e. a SIM card to be inserted in a mobile station. The smart card SC comprises means known as such which communicate with each other and which are intended to control the functions of the smart card SC and to implement data transmission (not shown in the figure). These means comprise, for example, a control unit (CPU) for controlling the function of the smart card on the basis of a program code stored in a program memory (ROM), and in a data memory (EEPROM) it is possible to store various user-specific data. During the function of the smart card SC, the random access memory (RAM) can be used as a temporary storage location for data. A bus adapter (DATA-I/O) on the smart card SC adapts the communication device functioning as a device for reading the smart card SC, for example a mobile station **MS1**, **MS2**, to connection lines, and to a control and data line. Physically, the smart card SC is coupled to the contacts available in the mobile station typically via its electrical contacts **4**. The features and the purpose of the smart card SC can be set by storing an application software corresponding to the purpose of use, in the program memory of the card SC advantageously at the manufacturing stage of the card. However, new technologies that make it possible to download applications to the smart card are being developed. Also, the protocols related to the interface of the smart card SC in connection with data transmission are taken care of by the application software used.

Further referring to FIG. **1**, the mobile station MS1 and MS2, for example a mobile phone, also comprises (not shown in the figure) means, known as such, for establishing a data transmission connection to the mobile communication network, means for reading the data of the smart card SC, such as a SIM card, and for storing the data on the SIM card, a control unit (CU) for controlling the different functions of the mobile station, which control unit advantageously comprises a micro controller unit (MCU) and a control logic circuit, such as an ASIC circuit (Application Specific Integrated Circuit). The functions include for example controlling the display and reading the keypad. The control unit also contains a memory attached to it, such as a read-only memory (ROM) and a random access memory (RAM). The function of the mobile station is controlled by means of an application software, which is responsible e.g. for implementing the protocols used in data transmission. The function of the mobile station is prior art known by anyone skilled in the art, and thus it is not necessary to describe it in more detail in this context. Known devices include the Nokia 8110, 6110 and 3110 mobile phones. As is well known, there are also devices available which contain two different user interfaces, for example the user interfaces of a mobile phone and a PDA device (Personal Digital Assistant). One such known device is the Nokia 9000 Communicator.

The aforementioned WAP application protocol is used in the following specification as an example of data transmission protocols to clarify the method according to the invention which is the object of this specification. In the following specification, said WAP clients and WAP servers refer advantageously to the clients and servers applying the WAP application protocol in the communication network. It is, of course, obvious that within the scope of the claims, it is also possible to apply the invention in connection with another application protocol, wherein the WAP indication mentioned in this specification will be used to refer to the use of this application protocol.

With reference to FIG. **2***a*, the protocol stack **100** (WAP Protocol Stack) in an OSI layer model of an advantageous WAP compatible system contains the following layers listed downwards from the top layer:

an application layer **101** for the wireless application environment WAE, which also comprises functions for a browser, which functions comprise a wireless markup language (WML), a WMLScript command language, and WTA services and WTA interfaces for telephone functions and programming interfaces, as well as content formats necessary for presenting information,

a wireless session protocol (WSP) of a session layer **102**,

a wireless transaction protocol (WTP) of a transaction layer **103**,

a wireless transport layer security protocol (WTLS) of a security layer **104**, intended to be used in connection with a WAP transport protocol,

a transport layer **105** for data packets of a wireless datagram protocol (WDP), which data packets also contain address information of the destination and other necessary information in addition to the actual data,

different bearer services **106**, which include for example transmission of short messages, circuit-switched data transmission and packet-switched data transmission.

To describe the different functions, the layer model advantageously refers to an ISO/OSI layer model known as such. The upper layers (WAE, WSP, WTP, WTLS) of the WAP architecture are independent of the data transmission network used, but the WDP layer **105** has to be applied according to the data transmission method used at a time, for example on the basis of the GSM network or special requirements. The WAP protocol stack also allows other services and applications **107** to utilize the WAP stack by means of specified interfaces.

FIGS. **7***a* and **7***b* illustrate the above-described alternative methods for utilizing a local resource. However, these methods entail problems which the invention aims to eliminate. FIG. **7***a* presents the use of a fixed API interface **70** to utilize a smart card SC, for example to read the content of a file contained in the smart card. In this context, an advantageously standardized interface **72** of the smart card SC is utilized. FIG. **7***b* presents another alternative, in which the necessary functions **71**, advantageously references to the system functions of the smart card SC, are included in a WAE application **101** to utilize the smart card SC directly by means of the application **101**. By means of the invention, however, the aim is to utilize the functions and definitions of the protocol stack **100** presented in FIG. **2***a*.

The purpose of the data transmission network and system is to provide the clients and servers located in the network with a communication channel, wherein the protocol stack **100** is utilized at the same time according to FIG. **2***b* in a way known as such. FIG. **2***b* presents the described logical structure of the client/server hierarchy, which comprises a client **1** located in a mobile station MS and a server **3** located in a communication network. WAE applications **101** (i.e. clients **1**) make a request to the WSP layer **102** by means of available command primitives. The layer **102** in question transmits the request further to the server **3**.

FIG. 3 presents the application of the invention, wherein the server 3 (i.e. the interface 4 of the smart card) is located in the same device MS with the client. Logically, the server 3 seems to be located in the communication network, wherein the request made by the client 1 is conducted according to the invention as if the server in question was a conventional server. According to the invention, the protocol layers are responsible for transmitting the request to the smart card SC interface 4. WAE applications 101 use the service primitives provided by the WSP layer 102 to transmit requests to the server 3. The server 3 processes the request and transmits a response to the client 1 (i.e. to the application 101). The primitive types of these service primitives can be divided for example into following types known as such:

the request conducted by an upper layer to obtain services from a lower layer,

the indication by means of which the layer providing services notifies an upper layer e.g. of a request made by a client or an activity initiated by the protocol,

the response, by means of which an upper layer notifies a lower level that it has received an indication,

the confirmation, by means of which the layer providing services confirms the maker of the request that the function is completed successfully.

The primitive types describe logical data transmission between adjacent layers in an abstract manner, and, unlike the API interface, they are not used to describe the actual implementation in detail.

According to FIG. 3, the use of two protocol stacks 100 in the same device MS is not necessary when implementing the invention, but it is possible to use one protocol stack 100 according to FIG. 4. According to the first preferred embodiment of the invention, it is possible to notice in the WSP layer 102 that the URL address, i.e. the network address, used by the application 101, advantageously a browser, refers to the smart card SC, wherein requests can be addressed directly to the smart card interface 4. With reference to FIG. 5, according to the second preferred embodiment of the invention, the interface 4 of the smart card can be logically located above the WDP layer 105, wherein the port numbers included in the request can be used to separate the smart card SC and the applications 101 of the communication device from each other. Thus, a separate port number i.e. a port is allocated for the interface 4 of the smart card.

The interface 4 of the smart card can also be logically located above the WTLS layer 104, wherein it is possible to utilize the functions of the security layer 104 in request transmission. Correspondingly, the interface 4 of the smart card can also be located above the WTP layer 103, especially in the case of connection-oriented data transmission. In both aforementioned cases, a separate port number is allocated for the interface 4 of the smart card.

Further referring to FIG. 5, according to a preferred embodiment of the invention, the WAE application 101 transmits a request to a local resource, advantageously to the smart card SC, by using the service primitives of the WSP layer 102. Correspondingly, the interface 4 of the smart card conducts the request to the smart card SC. The smart card SC gives a response to the interface 4 of the smart card, which generates a necessary response and transmits it to the WAE application 101. In more detail, the smart card SC is defined as a new server 3, which responds to requests related to the smart card SC. Thus, with respect to the WAE application 101, the smart card SC is like any server 3 located in the communication network, wherein logical transaction corresponds to the normal transaction of the protocol stack 100 according to FIG. 3. Requests are not, however, always directed to the communication network, but lower layers direct the requests to the interface 4 of the smart card, which can be located physically in the same device (MS) with the WAE application.

The WSP layer provides means for exchanging the information content between the applications of the client 1 and of the server 3. The services and protocols defined (WSP/B, Wireless Session Protocol/Browsing) are adapted especially for browser type applications. The WSP/B contains protocols both for a transport service of packets (datagram) in connectionless data transmission and for a transaction service of the session service in connection-oriented data transmission. The WSP/B follows closely the definitions of the HTTP data transmission protocol and supports applications which are HTTP/1.1 compatible. For example, methods such as GET, PUT and POST used by the HTTP can be used to retrieve (GET) or to transmit (PUT, POST) information. The header fields of the HTTP protocol can be utilized for giving information related to the content type of the message. It is also possible to use the header of the HTTP protocol for authentication. Correspondingly, the methods of the WWW communication network for authorization and data transmission can be utilized efficiently.

The invention is implemented in the WSP layer in such a way that the data transmission protocol used is deduced from the URL address, for example in this context "sc://", referring to the smart card, and the server used, described by the "<host>" part of the URL address. The requests whose address information begins with a reference "sc:///", are, according to the invention, directed to the interface of the smart card. According to the first preferred embodiment of the invention, and with reference to FIG. 4, this is effected via the WSP layer 102, and according to the second preferred embodiment of the invention and with reference to FIG. 5, this is effected via the WDP layer 105.

According to a preferred embodiment of the invention, the URL address to be used is formulated in the following way when the local resource is a smart card:

sc://<host>:<port>/<url-path>,

in which the "sc://" part refers to the smart card and the "<host>" part to the device to which the smart card is connected. When the "<host>" part is omitted, it can be assumed that the address refers to a smart card connected physically to the same mobile station. The "<port>" part can also be omitted, if a default port is used. Since a mobile station can simultaneously contain various smart cards, which, in turn, contain several different files which one wishes to browse or retrieve information from, these are separated from each other by means of the "<url-path>" part, for example in the following way:

sc:///sim/_7F2A/_6FO5.

In the following, an implementation of the invention according to FIG. 5 will be described, in which the smart card interface 4 is located above the WDP layer 105. The WDP layer 105 is located above bearer services 106 which are supported by different communication networks (GSM, GSM GPRS, etc). In this context, for example the communication system according to FIG. 1 will be utilized in a way known as such. The WDP layer 105 provides services for upper layers 101–104, which can thus be in transparent communication via the available bearer services 106. By means of the used port number, an upper layer entity is identified, which can be a WTP transport protocol 103, a WSP session protocol 102 or an application 101.

FIG. 6 presents a command sequence for using a local resource, especially a smart card SC, by means of a WAP application **101** (browser) particularly in connectionless data transmission. In the example, a request is made by means of a GET method to a SIM card **4** coupled to a mobile station MS. By means of a user interface (UI) **5**, the user enters (stage **201**) in a WAE application **101**, which is e.g. a browser, a determined URL address **6**, in which the "sc:///" part refers to the smart card SC, i.e. SIM card **4**, coupled to the mobile station MS in question. The address entered by the user has, for example, the format "sc:///sim/SomeApplication/SomeFile". At the next stage **202**, the browser **101** calls the service primitives of connectionless data transmission in the WSP layer **102**, in order to transmit a GET request to the smart card interface **4**. Here, the WSP layer **102** uses (stage **203**) a corresponding command primitive of the WDP layer **105**, in order to transmit a WSP/B request to the smart card interface **4**. A destination address used in the request refers to the mobile station MS itself, and a destination port number refers to a default port number allocated for the smart card interface **4**. The request used in FIG. **6** (stage **203**) is an service primitive T-DUnitdata, known as such, which is used to transmit data in a datagram, and by means of which it is possible to transmit parameters describing the destination address, the destination port, i.e. the address of the application **101** connected to the destination address, the source address, the source port, and user data transmitted by the WDP protocol **105**. In the request, all this information has to be given, but in an indication, the destination address, the destination port and the user data are advantageously sufficient. The destination address can also be an individualizing MSISDN number, an IP address, an X0.25 address, or another identification, known as such. The parameters are transmitted in a way known as such in packet transmission, wherein the packet contains for example header information and data, which information is coded in the packet into bit sequences of fixed size, typically octets, which are transmitted by means of the data transmission method used.

With reference to FIG. **6**, at the next stage, the WDP layer **105** detects that the destination address belongs to a local mobile station MS, i.e. to the wireless communication device MS, to which the smart card SC is coupled as well, wherein the indication is transmitted (stage **204**) to the appropriate port in the smart card interface **4**. The smart card interface **4** processes the indication and processes the WSP/B request by making the necessary requests to the smart card SC (stage **205**). After this, the smart card SC gives the content of the desired file to the smart card interface **4** (stage **206**), after which the interface **4** encapsulates the content in the WSP/B response and transmits (stage **207**) it as a request to the WDP layer **105**. At the next stage **208**, it is detected in the WSP layer **102** that the destination address and the destination port of the request belong to the browser **101** of the local mobile station MS, and the indication is transmitted (stage **208**) to the WSP layer. After this, the WSP layer **102** transmits (stage **209**) a WSP/B response to the browser **101**, and the browser **101** presents the content of the file to the user (stage **210**), advantageously by using the user interface **5**. The aforementioned WSP/B requests and responses can contain necessary header information, related, for example, to the content type of the message and to the authentication and authorization of the user. Furthermore, they can include data relating to the compression method used and data for parity checking.

In the above-described example, the user activates the use of local resources by giving a URL address by means of the user interface of the mobile station, which user interface is implemented in a way known as such by utilizing the display and keypad of the mobile station. In a preferred embodiment of the invention, the application itself can effect the activation, typically in order to retrieve information stored in the memory of the SIM card. Furthermore, it is obvious that according to the invention, also a server coupled to the communication system can request information from the smart card coupled to the mobile station. FIG. **8** illustrates the invention further in its third preferred embodiment, wherein a local resource, for example a smart card SC, receives requests also from clients **1** (i.e. from browsers **101**) other than those located in the same mobile station MS with the smart card SC. In this case, the interface **4** of the smart card operates like a server located in a network. In these different cases, the method already presented in FIG. **6** will be applied, which method can be implemented by anyone skilled in the art on the basis of the presented example. In this context, it has to be noted that the method also utilizes bearer services **106** according to FIG. **8** to transmit a request between mobile stations. The communication network available is also utilized therein.

The present invention is not restricted solely to the above described examples, but it can be modified within the scope of the appended claims. For example, the presented protocol stack can be implemented in a wireless communication device in which data transmission or the communication system is based on the previously mentioned IR, LPRF, SDRF data transmission, and in which the used bearer service is adapted for this data transmission.

What is claimed is:

1. A method in a communication system, which system (N, PLMN, PSTN, PDN) is arranged for transmitting information, requests (REQUEST), between a first mobile station (MS, MS1, MS2), a second mobile station .(MS, MS1, MS2) and a server (**3**, SERVER) connected to the communication system (N, PLMN, PSTN, PDN), wherein at least the first mobile station (MS, MS1, MS2) and at least the second mobile station (MS, MS1, MS2) comprise protocol means (**100**–**106**) for generating the request (REQUEST) and directing it to the communication system (N, PLMN, PSTN, PDN), which request (REQUEST) contains at least address information (URI, URL, URN) for identifying the destination of the request (REQUEST), and wherein at least the first mobile station (MS, MS1, MS2) comprises a first local resource (SC, **4**) coupled to the same, and in which method:

the request (REQUEST) is transmitted by the first mobile station (MS, MS1, MS2), the second mobile station (MS, MS1, MS2), or the server (**3**, SERVER),

characterized in that:

the address information (URI, URL, URN) of the request (REQUEST) is generated to identify said first local resource (SC, **4**), and

that the request (REQUEST) for said first local resource (SC, **4**) is generated and directed at least partly with said protocol means (**100**–**106**).

2. The method according to claim **1**, characterized in that:

the request (REQUEST) is generated in the second mobile station (MS, MS1, MS2), or in the server (**3**, SERVER),

that the first local resource (SC, **4**) coupled to the first mobile station (MS, MS1, MS2) is selected as the destination for the request (REQUEST), and

that the request (REQUEST) is transmitted at least partly by means of the communication system (N, PLMN, PSTN, PDN).

3. The method according to claim **1**, characterized in that:

the request (REQUEST) is generated in the first mobile station (MS, MS1, MS2), that the first local resource (SC, **4**) connected to the first mobile station (MS, MS1, MS2) is selected as the destination for the request (REQUEST), and

that the request (REQUEST) is transmitted and directed by with the protocol means (**100–106**) of the first mobile station (MS, MS1, MS2).

4. The method according to claim **1**, characterized in that the request (REQUEST) is generated at least partly by a local resource (SC, **4**) connected to the second mobile station (MS, MS1, MS2), or to the server (**3**, SERVER).

5. The method according to claim **1**, characterized in that address information (URI, URL, URN) identifying the local resource (SC, **4**) is entered into the application (**101**, BROWSER) by the user, to retrieve information from said local resource (SC, **4**), which application (**101**, BROWSER) is provided in the mobile station (MS, MS1, MS2) advantageously to present information for the user by means of a user interface (UI) of the mobile station (MS, MS1, MS2), and which application (**101**, BROWSER) is connected to the protocol means (**100–106**) to transmit said address information (URI, URL, URN).

6. The method according to claim **1**, characterized in that the request (REQUEST) is transmitted advantageously to receive an indication (INDICATION) from the local resource (SC, **4**), which indication (INDICATION) is directed at least partly with the help of said protocol means (**100–106**) to the mobile station (MS, MS1, MS2) or server (**3**, SERVER) which made the request (REQUEST).

7. The method according to claim **1**, characterized in that a WAP application protocol is applied to establish the protocol means (**100–106**).

8. The method according to claim **7**, characterized in that a connection is established at least from a WSP layer (**102**), a WTP layer (**103**), a WTLS layer (**104**), or a WDP layer (**105**) to an interface (**4**, SC INTERFACE) of said local resource (SC, **4**), which interface (**4**, SC INTERFACE) is connected to said local resource (SC, **4**), and which WSP layer (**105**), WTP layer (**103**), WTLS layer (**104**), or WDP layer (**105**) forms at least a part of the protocol means (**100–106**) of the WAP application protocol.

9. The method according to claim **1**, characterized in that a URI address, a URL address, or a URN name is used as address information.

10. The method according to claim **1**, characterized in that a smart card, such as a SIM card, connected to the mobile station (MS, MS1, MS2) is used as a local resource (SC, 4).

11. A communication system, which system (N, PLMN, PSTN, PDN) is arranged for transmitting information requests (REQUEST) between a first mobile station (MS, MS1, MS2), a second mobile station (MS, MS1, MS2) and a server (**3**, SERVER) connected to the communication system (N, PLMN, PSTN, PDN), wherein these comprise protocol means (**100–106**) for generating a request (REQUEST) and directing it to the communication system (N, PLMN, PSTN, PDN), which request (REQUEST) contains at least address information (URI, URL, URN) for identifying the destination of the request (REQUEST), and wherein at least the first mobile station (MS, MS1, MS2) comprises a first local resource (SC, **4**) connected thereto, characterized in that:

said first local resource (SC, **4**) is arranged to be identified by means of the address information (URI, URL, URN) of the request (REQUEST),

that the request (REQUEST) addressed to said first local resource (SC, **4**) is arranged to be generated and directed to said first local resource (SC, **4**) at least partly with the help of said protocol means (**100–106**), and that the communication system (N, PLMN, PSTN, PDN) is arranged to transmit the request (REQUEST) addressed to said first local resource (SC, **4**) by the second mobile station (MS, MS1, MS2) and/or the server (**3**, SERVER).

12. A wireless communication device comprising protocol means (**100–106**) for generating a request (REQUEST) and directing it to a communication system (N, PLMN, PSTN, PDN), which request (REQUEST) contains at least address information (URI, URI, URN) to identify the destination of the request (REQUEST), and which wireless communication device (MS, MS1, MS2) advantageously comprises a local resource (SC, **4**) connected thereto, characterized in that:

the local resource (SC, **4**) is arranged to be identified by means of the address information (URI, URL, URN) of the request (REQUEST), and

that said protocol means (**100–106**) are arranged for directing the request (REQUEST), addressed to said local resource (SC, **4**), to said local resource (SC, **4**).

13. The wireless communication device according to claim **12**, characterized in that said protocol means (**100–106**) are arranged for directing and transmitting a request (REQUEST) received by said wireless communication device (MS, MS1, MS2), a request (REQUEST) generated in said wireless communication device (MS, MS1, MS2), or a request (REQUEST) according to both these alternatives, to said local resource (SC, 4).

14. The wireless communication device according to claim **12**, characterized in that said protocol means (**100–106**) are arranged also to direct and transmit an indication (INDICATION) received as a response to the request (REQUEST) from the local resource (SC, **4**).

15. The wireless communication device according to claim **12**, characterized in that said protocol means (**100–106**) are also arranged for directing and transmitting the request (REQUEST) generated by said local resource (SC, **4**).

16. The wireless communication device according to claim **12**, characterized in that the local resource (SC, 4) is a smart card, such as a SIM card.

17. The wireless communication device according to claim **12**, characterized in that the address information (URI, URL, URN) comprises a URI address, a URL address, or a URN name.

* * * * *

US006233688B1

(12) **United States Patent**
Montenegro

(10) **Patent No.: US 6,233,688 B1**
(45) **Date of Patent: May 15, 2001**

(54) **REMOTE ACCESS FIREWALL TRAVERSAL URL**

(75) Inventor: **Gabriel Montenegro**, Fremont, CA (US)

(73) Assignee: **Sun Microsystems, Inc.**, Mountain View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/109,260**

(22) Filed: **Jun. 30, 1998**

(51) **Int. Cl.**^{7} **G06F 11/00**

(52) **U.S. Cl.** **713/201**

(58) **Field of Search** 713/201, 200, 713/202; 707/3, 4, 100, 102; 709/232, 237; 380/25, 29

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,818,019	*	6/1999	Valencia	395/200.57
5,822,539	*	10/1998	Van Hoff	395/200.66
5,944,823	*	8/1999	Jade et al.	713/201
5,950,195	*	9/1999	Stockwell et al.	707/4
5,987,611	*	11/1999	Freund	713/201
5,999,979	*	12/1999	Vellanki et al.	709/232
6,061,797	*	5/2000	Jade et al.	713/201
6,073,176	*	6/2000	Baindur et al.	709/227
6,088,796	*	7/2000	Cianfrocca et al.	713/152

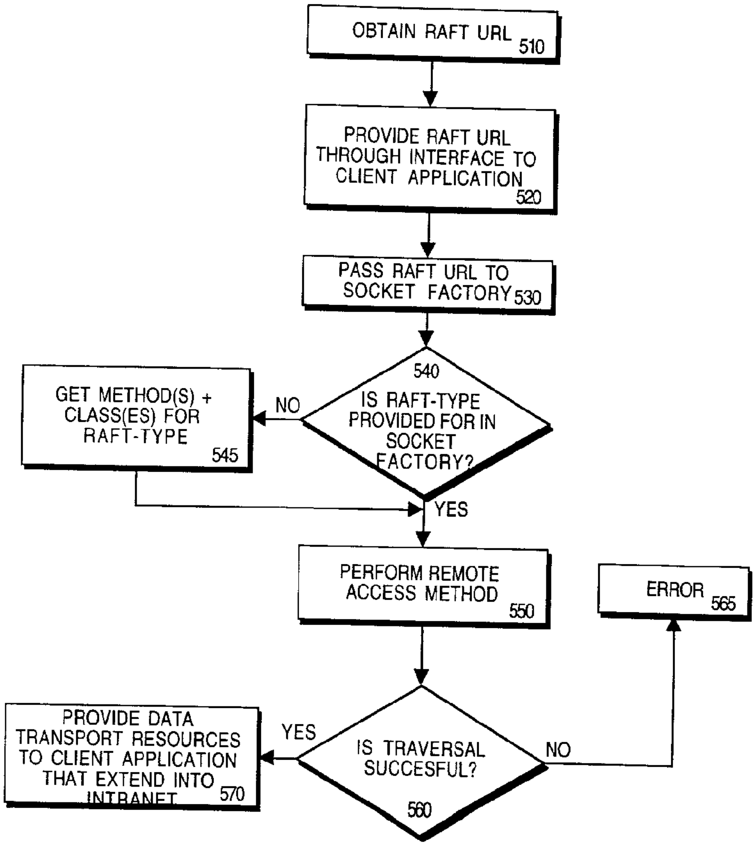
* cited by examiner

Primary Examiner—Nadeem Iqbal
(74) *Attorney, Agent, or Firm*—Blakely Sokoloff Taylor & Zafman

(57) **ABSTRACT**

The invention provides a generic naming scheme for remote access and firewall traversal in the form of a uniform resource locator (RAFT URL). The RAFT URL may be provided to any client, regardless of compatibility with the remote access/firewall traversal method, which then launches an operating environment code module. The operating environment code module performs the remote access/firewall traversal method and interacts with the operating environment to obtain data transport mechanisms. These mechanisms permit the client application to transact with private resources beyond the firewall. The remote access/firewall traversal procedure is made transparent to the client application, and thus, a wider array of client applications may be chosen for the data session with the resources beyond the firewall.

9 Claims, 5 Drawing Sheets


OBTAIN RAFT URL 510
PROVIDE RAFT URL THROUGH INTERFACE TO CLIENT APPLICATION 520
PASS RAFT URL TO SOCKET FACTORY 530
540 IS RAFT-TYPE PROVIDED FOR IN SOCKET FACTORY?
NO
GET METHOD(S) + CLASS(ES) FOR RAFT-TYPE 545
YES
PERFORM REMOTE ACCESS METHOD 550
ERROR 565
IS TRAVERSAL SUCCESFUL? 560
YES
NO
PROVIDE DATA TRANSPORT RESOURCES TO CLIENT APPLICATION THAT EXTEND INTO INTRANET 570

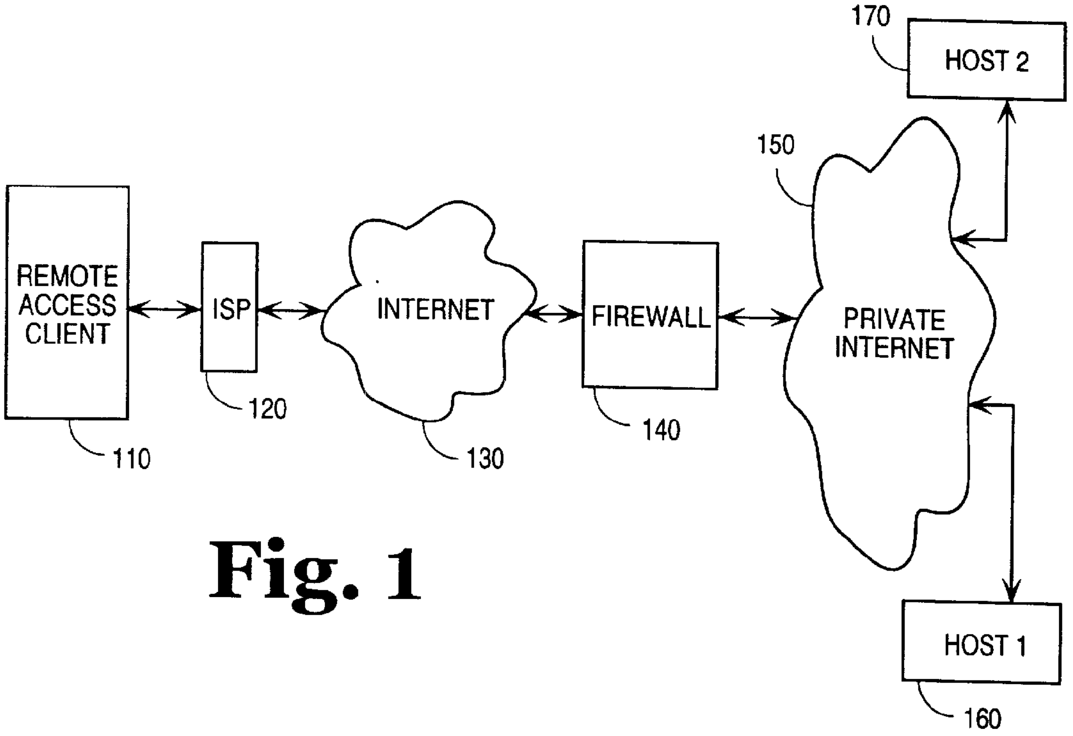

170
HOST 2
150
REMOTE ACCESS CLIENT
ISP
INTERNET
FIREWALL
PRIVATE INTERNET
120
110
130
140
HOST 1
160

Fig. 1

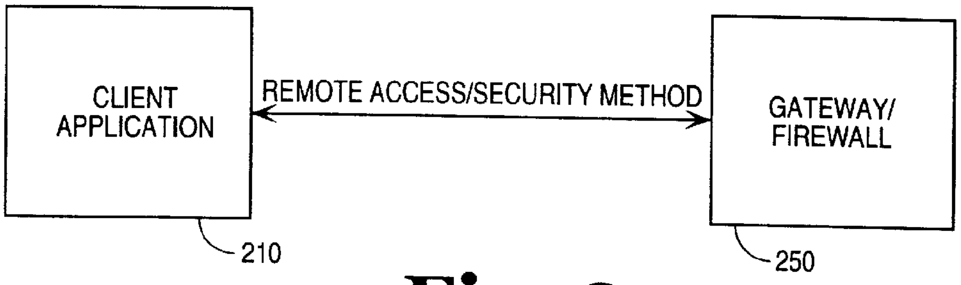

CLIENT APPLICATION
REMOTE ACCESS/SECURITY METHOD
GATEWAY/ FIREWALL
210
250

Fig. 2

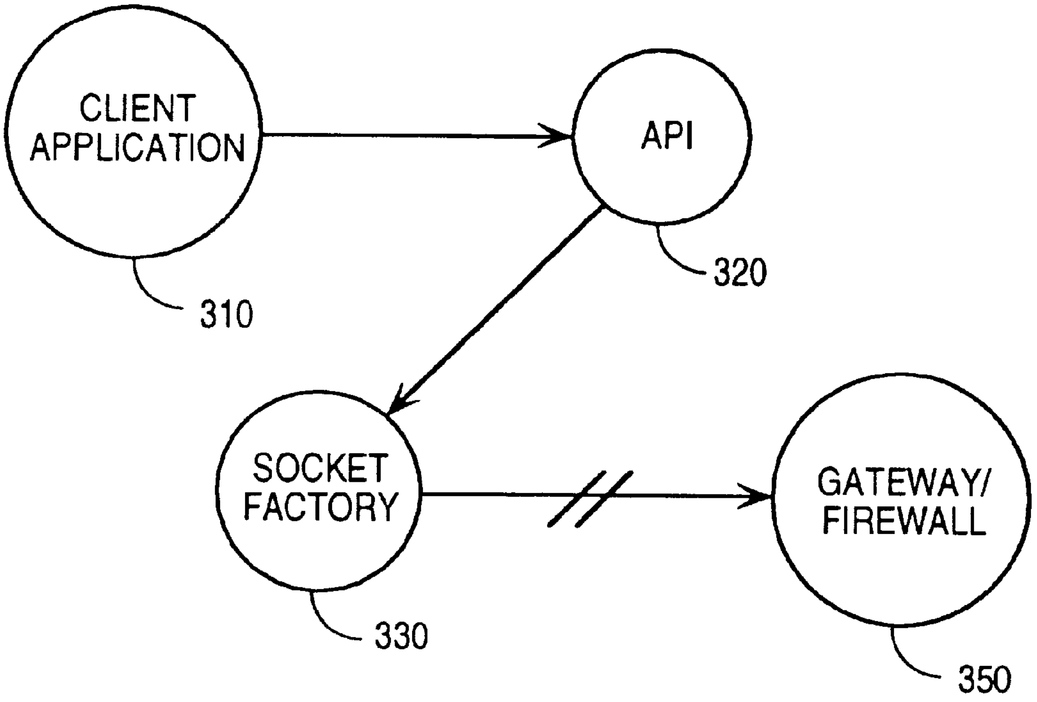

CLIENT APPLICATION
310
API
320
SOCKET FACTORY
330
GATEWAY/ FIREWALL
350

Fig. 3

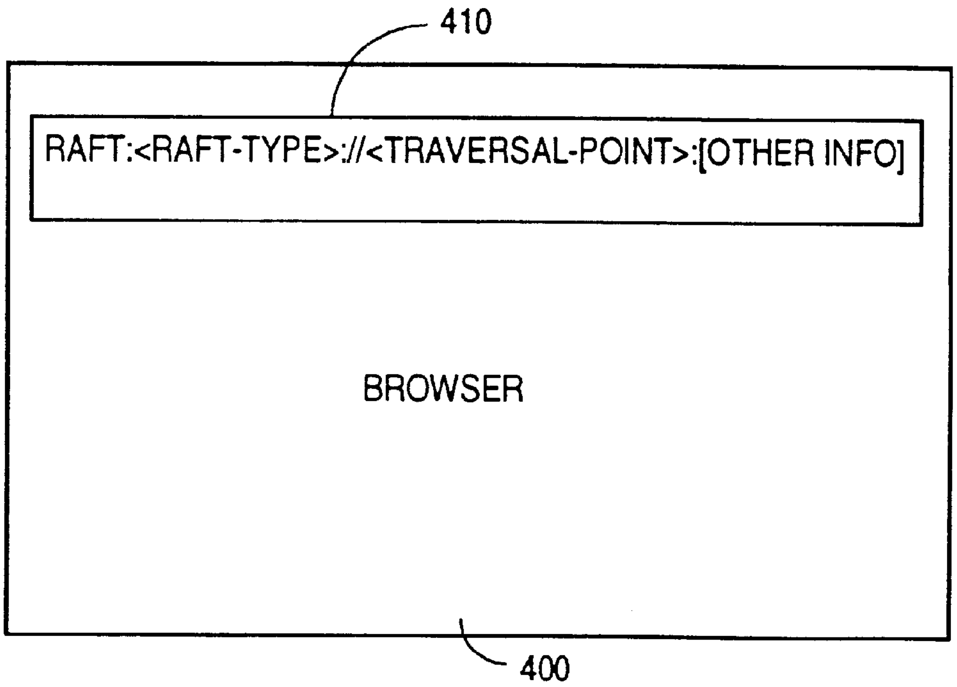

410
RAFT:<RAFT-TYPE>://<TRAVERSAL-POINT>:[OTHER INFO]
BROWSER
400

Fig. 4a

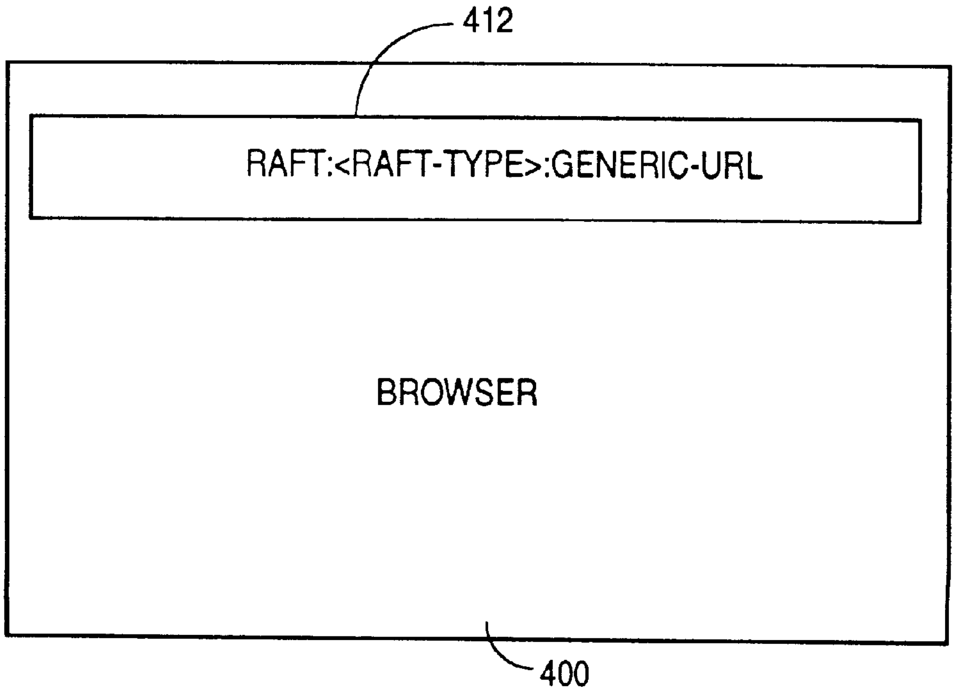

412
RAFT:<RAFT-TYPE>:GENERIC-URL
BROWSER
400

Fig. 4b

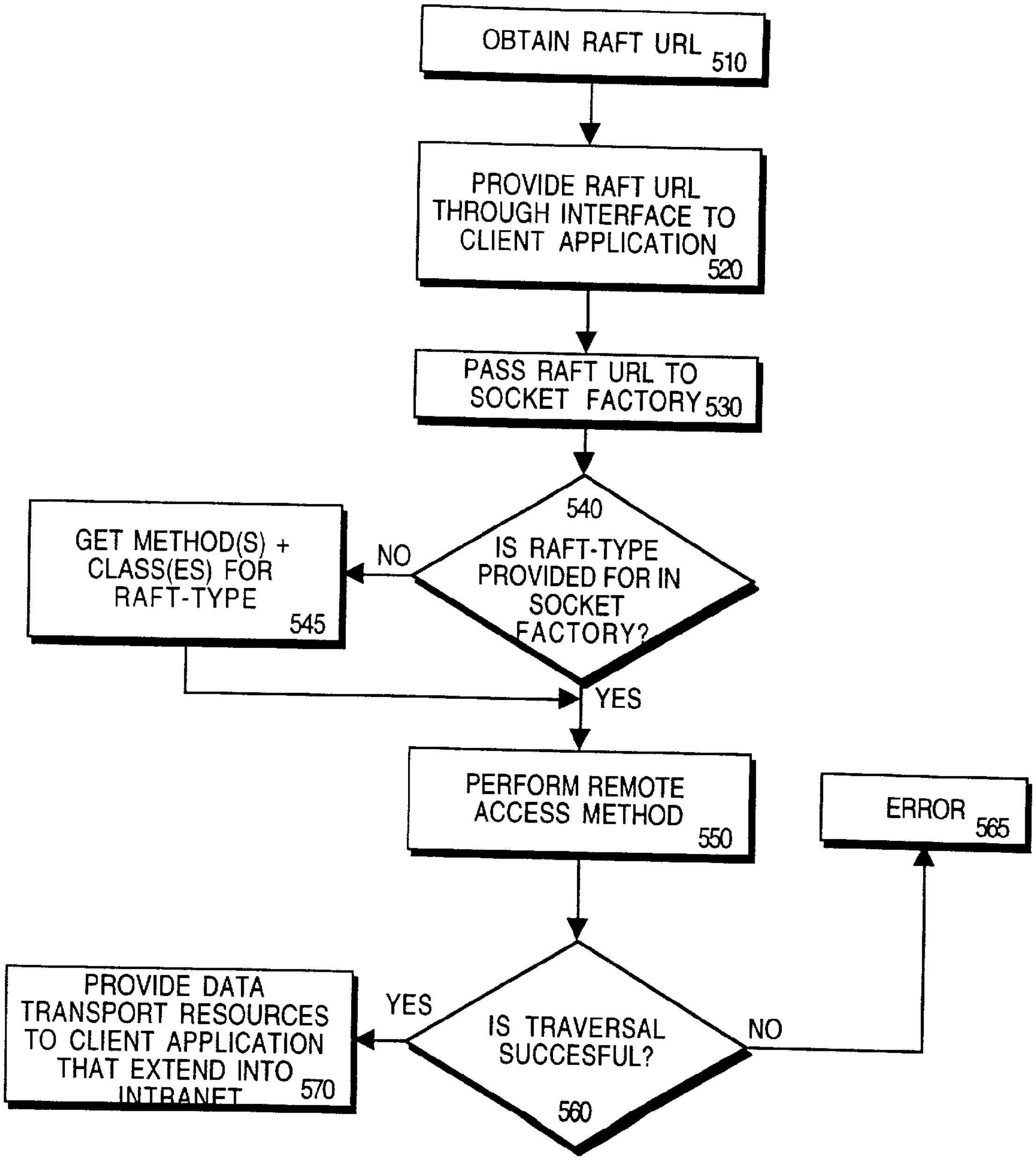

OBTAIN RAFT URL 510
PROVIDE RAFT URL THROUGH INTERFACE TO CLIENT APPLICATION 520
PASS RAFT URL TO SOCKET FACTORY 530
540
IS RAFT-TYPE PROVIDED FOR IN SOCKET FACTORY?
NO
GET METHOD(S) + CLASS(ES) FOR RAFT-TYPE 545
YES
PERFORM REMOTE ACCESS METHOD 550
ERROR 565
IS TRAVERSAL SUCCESFUL?
560
YES
NO
PROVIDE DATA TRANSPORT RESOURCES TO CLIENT APPLICATION THAT EXTEND INTO INTRANET 570

Fig. 5

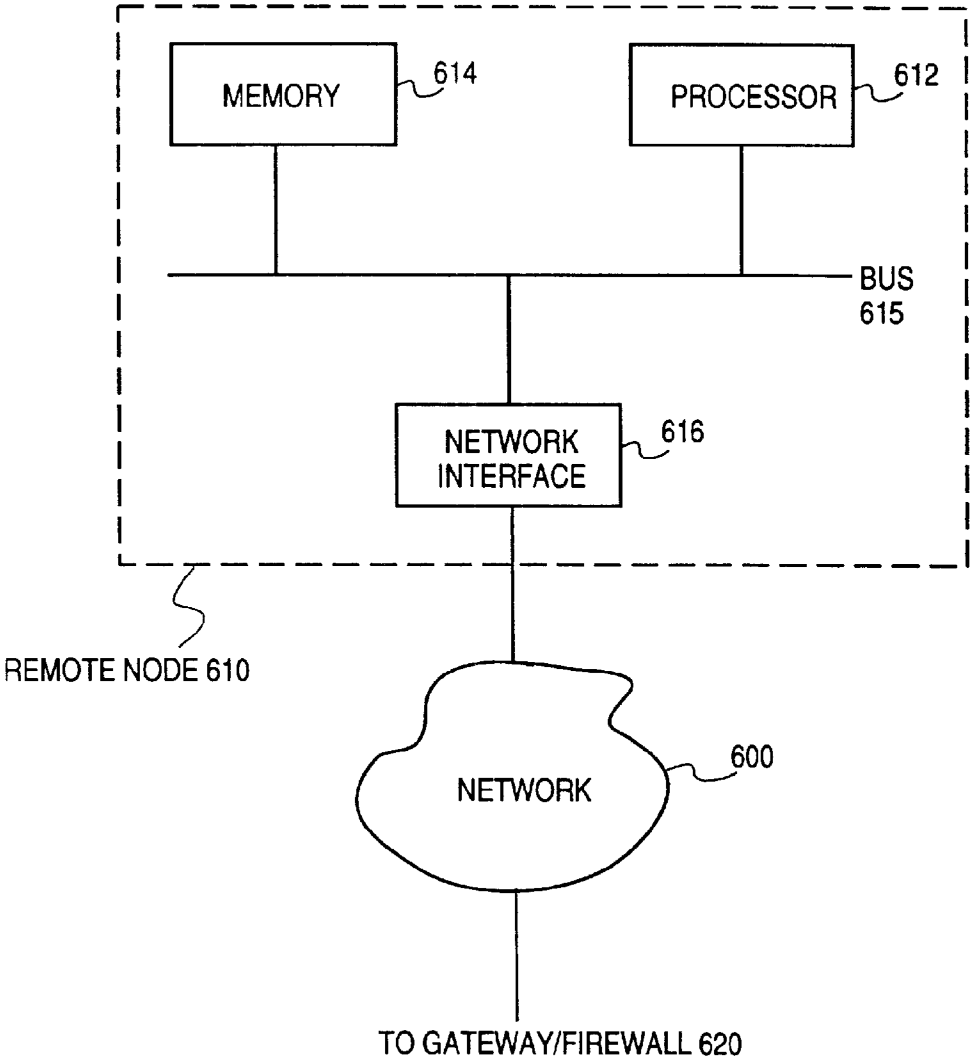

MEMORY
614
PROCESSOR
612
BUS
615
NETWORK
INTERFACE
616
REMOTE NODE 610
NETWORK
600
TO GATEWAY/FIREWALL 620

Fig. 6

REMOTE ACCESS FIREWALL TRAVERSAL URL

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention relates generally to the field of computer networking. More specifically, the invention relates to protocols and conventions for computer inter-networking.

2. Description of the Related Art

From within a corporate "intranet" network or a shared private network, the methods and protocols for local area access to computers, devices and data resources within the network is well defined and somewhat uniform, under the control of the administrators of those networks. When users attempt to gain access to those same devices, computers and resources from outside the network, such access is referred to as "remote access". In the past, the most popular physical topology for remote access is a direct dial into a modem bank, which may be at the corporate site or provided for by an ISP (Internet Service Provider). However, this topology impose a heavy administrative burden and monetary cost especially when remote access is attempted through long distance or international toll calls. Thus, there has been a recent trend to provide remote access through an Internet connection. With Internet remote access, the IP (Internet Protocol) connection can be obtained first using any available method, and thus the intranet does not need to maintain a direct physical access point such as a dial-in modem bank. Once a user is "on the Internet" (has achieved an IP connection), a multitude of different protocols and services (limited by the connectivity features of the intranet) can be used on the user's "client" to gain remote access into the intranet. In order to gain remote access, the client must pass the intermediary step, in most cases, of traversing a firewall. The traversal of the firewall can be achieved by using gateway specific software, SSL (Secure Sockets Layer) mechanisms and so on.

Specific client software must have support and awareness of specific firewall traversal methods, and thus generic client software cannot be utilized to penetrate the intranet. For example, a client application such as Netscape™ may not be able to traverse the firewall since it lacks the means with which to express entry parameters to "support" the private intranet's firewall scheme. Thus, users are often limited to using software that specifically understands and communicates with the intranet. This restricts the choice of client software greatly such that only a limited set of client applications out of all the multitude of programs available can be used when accessing that private intranet.

These schemes typically tie the firewall access mechanism to the application, instead of making it transparent by placing it inside the underlying networking support. There is a need for general naming mechanism in order to separate application from firewall traversal mechanisms. Furthermore, the firewall has no standard format to download traversal configuration into the client after authentication.

Thus, there is a need for a generic scheme for allowing client applications to be transparent to the remote access and firewall traversal procedure. The scheme should permit any type of remote access/firewall traversal and security method/protocol to be recognized and operated independent of the client application.

SUMMARY OF THE INVENTION

The invention provides a generic naming scheme for remote access and firewall traversal in the form of a uniform resource locator (RAFT URL). The RAFT URL may be provided to any client application, regardless of compatibility with the remote access/firewall traversal method, which then launches another executable module. The executable module performs the remote access/firewall traversal method and interacts with the operating environment to obtain data transport mechanisms. These mechanisms permit the client application to transact with private resources beyond the firewall. The remote access/firewall traversal procedure is made transparent to the client application, and thus, a wider array of client applications may be chosen for the data session with the resources beyond the firewall.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects, features and advantages of the method and apparatus for the present invention will be apparent from the following description in which:

FIG. **1** illustrates the topology of remote access to a private intranet.

FIG. **2** illustrates a traditional remote access scheme.

FIG. **3** is a logical diagram of one embodiment of the invention.

FIG. **4***a* illustrates a RAFT URL according to one embodiment of the invention.

FIG. **4***b* illustrates a RAFT URL according to another embodiment of the invention.

FIG. **5** is a flow diagram of RAFT URL processing according to one embodiment of the invention.

FIG. **6** is a diagram of one embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

Referring to the figures, exemplary embodiments of the invention will now be described. The exemplary embodiments are provided to illustrate aspects of the invention and should not be construed as limiting the scope of the invention. The exemplary embodiments are primarily described with reference to block diagrams or flowcharts. As to the flowcharts, each block within the flowcharts represents both a method step and an apparatus element for performing the method step. Depending upon the implementation, the corresponding apparatus element may be configured in hardware, software, firmware or combinations thereof.

FIG. **1** illustrates the topology of remote access to a private intranet.

Internet-based remote access may involve many different entities which intervene between a remote access client **110** and hosts such as host **160** and host **170**. Remote access client **110** may consist of a computer system or PDA (Personal Digital Assistant) or other information device that has the capability of being connected to a network. Remote access client **110** may also execute software configured to enable the access of services such as FTP (File Transfer Protocol), POP (Post Office Protocol), HTTP (HyperText Transfer Protocol), etc. One way of "getting on the Internet" to commence the remote access session with one or more hosts **160** and **170** is to dial-up to an ISP (Internet Service Provider) whose internal routers and switches provide a TCP/IP (Transport Control protocol/Internet Protocol) connection to remote access client **110**. With the "Internet connection thus obtained, the remote access client **110** can reach any computer connected to the "Internet" **130** which is well-known in the art. The ISP dial-up method of Internet access described above is merely exemplary of many ways a connection to the Internet may be obtained.

Once on the Internet **130**, remote access client **110** is free to access services provided by any computers or networks connected to the Internet **130** such as an HTTP service (i.e., a web site). However, the below description restricts itself to using the Internet to gain remote access into a private network or intranet **150** by means of the Internet. Once remote access into the intranet **150** is established, remote access client **110**, services provided for within the intranet **150** may be accessed. What distinguishes the intranet **150** from an ordinary web server or FTP server on a more public network is the security and isolation that can be provided by a gateway or firewall **140**. One function of firewall **140** is to prevent unauthorized access into the intranet by users/computers connected to the Internet **130**.

To control and configure access to the intranet **150** through firewall **140** many various firewall protections or security schemes have been developed such as IP security schemes using SKIP (Simple Key Management for Internet Protocols), or ISAKMP (Internet Security Association and Key Management Protocol), SSL (Secure Sockets layer), etc. Many of these schemes are conflicting and standardization has not been successfully achieved. One reason for the failure of standardization is the nature of remote access—it is intended for a specific often closed set of users. For instance, a company X may desire that only certain key employees have remote access to the intranet of X. In that case, company X will choose whatever method of remote access security is easy to implement or whatever method is decided on as best for the type of information served. Since the choice of remote access security methodology is isolated to the company implementing it, standardization is difficult. In a more public network such as the Internet standardization is easy to achieve since the many nodes of the network desire compatibility with the other.

Lack of standardization limits the remote access client's **110** choice of remote access security methods when using software to access services provided by the intranet.

The invention in its various embodiments permits the use of a wider range of client software to access the intranet **150**. A naming scheme is provided which when used by client software will cause the client **110** to negotiate remote access. The naming convention is generalized so that any of the conventional security methods can be adequately identified. In one embodiment, the client software parses a RAFT URL (Remote Access Firewall Traversal Uniform Resource Locator) and executes code that extends the ability of the underlying operating system of remote access client **110** to negotiate access according to the RAFT URL. The invention in certain embodiments makes transparent the process of firewall traversal. This transparency will allow the use of application software that is not restricted by or concerned with the remote access security and traversal method.

Currently, there is no transparency between the procedure of firewall traversal for remote access and the client application software used to thereafter access the services provided by the intranet beyond the firewall, if the traversal method uses application layer traversal mechanisms. FIG. **2** illustrates this prior art model of application dependency upon the remote access security method. A client application **210** is executed on a system somewhere outside in a topology sense from the firewall **250** and intranet.

In traversal methods like httpstunneling, client application **210** must be an application specifically aware and capable of the remote access security and traversal method. The particulars of this method are predefined by the gateway **250** and cannot be modified. As such, if an application such as a web browser does not have built-in support for the remote access method, it will be unable to gain remote access. The process of attaining remote access and/or traversing a firewall is thus, within the purview of the client application in application layer transversal mechanisms.

The RAFT URL mechanism of the invention provides benefits application or session layer traversal mechanisms that imply client software modifications. The RAFT URL provides a universal language to denote what these modifications are. It promotes the standardized use of these mechanisms as a consistent set of functions that become a service to client applications rather than an inherent part of them.

For firewalls in general, RAFT URLs provides a universal language that aids in unifying firewall technology. Currently, some firewalls are based on session layer or application layer mechanisms (application layer gateways or ALGs), whereas other firewalls operate at lower layers, in a much more transparent (to applications) fashion, i.e., IPSEC firewalls, network layer tunneling firewalls. RAFT allows these firewall technologies to be treated in a similar fashion, and to allow their integration and tracking within one single mechanism.

FIG. **3** is a logical diagram of one embodiment of the invention.

Unlike the prior art model of FIG. **2**, the client application **310** does not have the responsibility of directly securing remote access. Rather, the client application **310** makes use of a socket factory **320** (Application Program Interface) to access a system resource such as sockets. The socket factory **320** negotiates security and access to gateway/firewall **350**. The socket factory is configured to understand the naming convention (RAFT URL) and then initiate steps to obtain remote access, which includes the negotiation of security protocols defined by the RAFT URL. The RAFT URL is input either by user or by preference setting to the client application **310**. The socket factory recognizes the RAFT URL (see FIG. **5**) and configures itself **330** to communicate with gateway/firewall **350**.

This socket factory **320** can be made available to any other client application as well, such as a client application **315** or **312**. Like client application **310**, client applications **312** do not need to be compliant or compatible with the firewall security method. When client applications **312** or **315** obtain sockets via the socket factory, these applications will inherit the same behavior as provided in the sockets to transmit/receive information from gateway/firewall **350**. Once the socket factory has successfully gained access beyond the firewall, the sockets derived from the socket factory **330** provide presentation layer data transport mechanisms to client application **310**.

Client application **310** (or **312** or **315**) would be unaware, except for the obtaining of behavior from the sockets of the firewall traversal. Once client application **310** has obtained the right to transact via the sockets mechanism to gateway/firewall **350**, other applications **312** and **315** can use sockets in a similar fashion to communicate beyond the firewall **350**. The traversing of the firewall is divorced from the client application **310** and made the responsibility of the socket factory **320**. In so doing, traversal of the firewall and its security method is made transparent to client applications **310, 312** and **315**. Unlike the prior art of FIG. **2**, this permits the client application to be less specific to the firewall.

FIG. **4***a* illustrates a RAFT URL according to one embodiment of the invention.

The RAFT URL is a naming scheme generic to the various firewall traversal and remote access security methods. Whatever method the particular firewall designates, the RAFT URL has a structure that can contain identifiers for the underlying implementation (socket factory in the Java case) to handle its negotiation.

FIG. **4***a* shows a system application **400** like one used to configure the socket factory which accepts as input or can store as preference a RAFT URL **410**. Unlike an ordinary URL, the RAFT URL does not point to a data object or data creator. Rather, it designates the means to negotiate firewall access which includes specifying security methods either implicitly or explicitly. In one embodiment, the RAFT URL **410** has the following components:

"raft"—indicates the identifying information to follow are handles to configure remote access. Unlike "http:" or "ftp:" of an ordinary URL which identifies a data transfer protocol or service, the "RAFT:" designation can serve to launch or call the remote access mechanism.

"raft-type"—designates the particular name given to a specific firewall traversal or remote access method. For instance, the use of layer **3** tunneling with SKIP (Simple Key-Management for Internet Protocols) or tunneling through the firewall using SSL.

"traversal point"—indicates the IP address, domain name or other location of the firewall, gateway, or remote access server with which the client system must negotiate access, the traversal point will be known to an authorized user or can be provided through some other means through authentication.

"other-info"—indicates a security scheme specific initialization string such as a password, user name or even a secondary security mechanism. The parameter is optional with its format defined wholly by the scheme and firewall's policy.

"generic-URL"—allows regular URL's of the "http:" or "ftp:" variety.

FIG. **4***b* illustrates a RAFT URL according to another embodiment of the invention.

Like FIG. **4***a*, both the designation of "raft:" and a raft-type are provided. Instead of a traversal point, a generic-URL parameter terminating it. Such an embodiment for the RAFT URL may be used in https or secure http protocols, where the "https" URL designates the entry point through the firewall.

Implementation for Specific Schemes

Described in a listing of desirable implementations of the RAFT URL for specific remote access and firewall traversal schemes. The RAFT URL concept presented in its various embodiments permits any type of scheme to be designated.

1. SSL Tunneling

The RAFT URL for SSL tunneling would take the form "raft:sslt:https://x" where x may designate a host port or server address, directory path and search/cgi (common gateway interface) or file name parameters.

2. Mobile IP Firewall Traversal

The RAFT URL for firewall traversal by mobile IP (Internet Protocol) would take the form "raft:mip://traversal-point" with an optional terminating string ";type=y", where y refers to either the "SKIP" (Simple Key Management for Internet Protocols) security protocol or IP security protocols.

3. Remote Access Using TSP

The RAFT URL for remote access using TSP (Tunneling Set-up Protocol) takes the form "raft:tsp://x" where x includes a traversal point and optional ";type=y" terminating parameter. Y may be either "ipsec" or "skip" as defined for Mobile IP access in part 2 above.

For instance, consider the following RAFT URL" "raft:sslt:https://firewall.foo.com/access.html". This RAFT URL indicates that remote access through the firewall named/addressed as "firewall.foo.com" is to be achieved using secure http by processing the page "access.html" which may be an interactive login page where a user enters login information such as a user name and password for further entry beyond the firewall.

The RAFT URL is useful due to its versatility; any type of remote access may be designated and specified. A second feature of the invention lies in the processing of the RAFT URL in a manner transparent to the client data application.

FIG. **5** is a flow diagram of RAFT URL processing according to one embodiment of the invention.

The first step in creating a transparency between the firewall traversal for remote access and the client application is to obtain the appropriate RAFT URL (step **510**). The discovery of the specific RAFT URL to use is not a subject of the essential invention. Obtaining a RAFT URL may be achieved in several ways: (1) obtaining it in person from a system administrator, (2) visiting a special web page where an authenticated user may retrieve the appropriate RAFT URL from the firewall, (3) querying a directory service such as LDAP (Lightweight Directory Access Protocol) or (4) may be preconfigured into the client application or system. The appropriate RAFT URL will designate parameters allowing the client system to get private intranet resources through its data transport mechanisms (IP stack, sockets, etc.).

Assuming that the RAFT URL is obtained, it is then provided through some interface to the client application (step **520**). This interface may be URL data entry dialog of a browser or be chosen from a menu by the user. The RAFT URL may already be provided to the client application by being set-up in a preference manager for the client application or in an operating system registry intended to service that client application. When so provided, the RAFT URL is passed to the socket factory (in Java) (step **530**) that will execute methods needed to perform the firewall traversal procedure.

If the specific "raft-type" (type of firewall traversal and/or remote access security, see above) is not provided for in the socket factory, by way of methods, functions, classes and/or code that can be executed, then the required methods, functions, classes, code, etc. must be obtained. In that case, the firewall traversal procedure first gets the needed methods, classes, etc. for the raft-type. These methods, classes, etc. may be obtained from the firewall itself and then loaded as an API (Application Program Interface) or mobile code, such as Java applet, onto the client system. If the required methods, classes, etc., to undergo processing of the "raft-type" is available or made available then the remote access method is performed (step **550**). The remote access and/or firewall traversal method, as specified by the "raft-type" parameter of the RAFT URL, need not be known or compatible with the client application(s) ultimately handling the data transactions with resources beyond the firewall. This responsibility and restriction is removed to the socket factory.

If the traversal or remote access is successful (checked at step **560**), the data transport resources are provided to the client application that extend into the intranet (step **570**). The data transport resources may be sockets, a TCP/IP stack or other presentation/transport layer mechanisms which facilitate data transport between the client system and resources such as servers existing in the intranet beyond the firewall. Access is regulated by and monitored by the firewall. The interaction between the API/applet and the obtained data transport resource is dependent upon platform and operating system and will vary from system to system.

For example, consider a mobile network computer system based upon Java. Such a computer system typically uses a socket factory to create network data transport resources (called sockets). These sockets allow applications to transact data over a network such as the Internet. Currently, firewall traversal and remote access security operates above this transport layer on the application layer. As a consequence, the choice of client applications restricted when transacting data within an intranet beyond the firewall since compatibility is required. SSL attempts to generalize secure access but does so on the application level and thus, the client application too must be SSL compatible. Further, if SSL is not offered by the firewall as the secure access method, then SSL fails to be a general solution which will be adequate for every firewall traversal and remote access security situation. In the Java based mobile computer system, the invention, in one embodiment, would operate to set the socket factory according to parameters in the RAFT URL. Client applications that use the standard network resources Java application (known as java.net) can also be divorced from having to include the methods, functions, classes, etc. needed to perform remote access in accordance with the RAFT URL. Rather, a system applet would parse the RAFT URL and set the socket factory by adding methods, functions, classes (if not already present) and then allow the socket factory to handle the firewall traversal and remote access. If a client application wishes directly control its remote access security scheme, it may directly add the required behavior (methods, functions, classes, etc.) to configure its use of the socket factory provided for in the operating system. The actual negotiation of remote access or firewall traversal is made transparent to the application layer, which takes advantage of data transport resources after firewall traversal is successfully complete.

The above example refers to a Java-based mobile computing platform. However, the invention, in its various embodiments, is not limited to any platform or operating environment. An application or other network data transport resource can be provided with the means to accept and parse the RAFT URL and then carry out the identified remote access procedure. If the remote access behavior is not available to the operating system, it may be obtained automatically by the application.

FIG. **6** is a diagram of one embodiment of the invention.

A RAFT URL mechanism would allow a remote node **610** to traverse into gateway/firewall. Remote node **610** is illustrative of a computer system or information device that desires remote access to a private or other network that lies beyond the firewall. Remote node **610** may be connected to gateway/firewall via a network **600** such as the Internet.

Remote node **610** may be composed of a variety of devices, including a memory **614** and a processor **612** coupled directly to each other and through a bus **615**. Bus **615** may also attach a network interface card **616** to both memory **614** and processor **612**. Network interface card **616** connects the remote **610** to network **600** and allows remote node **610** to transact data to and from network **600** and consequently, to and from other nodes that may be connected to network **600** such as the gateway/firewall. A RAFT URL that identifies the firewall and its security access scheme may be provided to remote node **610** in a variety of ways, including transmission over network **600**. Once the appropriate RAFT URL is provided to remote node **610** the RAFT URL is passed to a socket factory generated by some application running in memory **614** and executed by processor **612**. The processor **612** is configured (by instructions provided thereto) to parse the RAFT URL and initiate the appropriate client events on remote node **610** for traversal of gateway/firewall **620**. Once remote access is granted to remote node **610**, it is free to transact data with resources beyond the firewall **620**. This data transaction is handled by network interface **616** which may modify data packets with any security-related headers or encapsulation demanded by the RAFT URL's designation.

The exemplary embodiments described herein are provided merely to illustrate the principles of the invention and should not be construed as limiting the scope of the invention. Rather, the principles of the invention may be applied to a wide range of systems to achieve the advantages described herein and to achieve other advantages or to satisfy other objectives as well.

What is claimed is:

1. A method of remote access comprising:

providing a remote access firewall traversal (RAFT) uniform resource locator (URL) to a client, said RAFT URL indicating a mode of remote access through a firewall; and

configuring said client to negotiate access to a private resource protected by said firewall based on parameters specified by said RAFT URL that allow said client to access said private resource through its data transport mechanisms.

2. A method according to claim **1** wherein the step of recognizing a RAFT URL includes the steps of:

identifying a RAFT service;

identifying a RAFT type, said RAFT type denoting a specific remote access method.

3. A method according to claim **2** wherein the step of recognizing a RAFT URL further includes the steps of:

identifying a generic uniform resource locator.

4. A method according to claim **2** wherein the step of recognizing a RAFT URL further includes the steps of:

identifying a traversal point.

5. A method according to claim **4** wherein the step of recognizing a RAFT URL further includes the steps of:

identifying a scheme-specific initialization string.

6. A method according to claim **1** wherein the step of configuring includes the steps of:

building sockets from a socket factory in accordance with said RAFT URL; and

passing the behavior of said sockets from said socket factory to application on said client, said application able to traverse said firewall with the appropriate method.

7. A method according to claim **6** further comprising the step of:

obtaining methods and classes for a raft-type specified in said RAFT URL if said raft-type is not provided for by said socket factory.

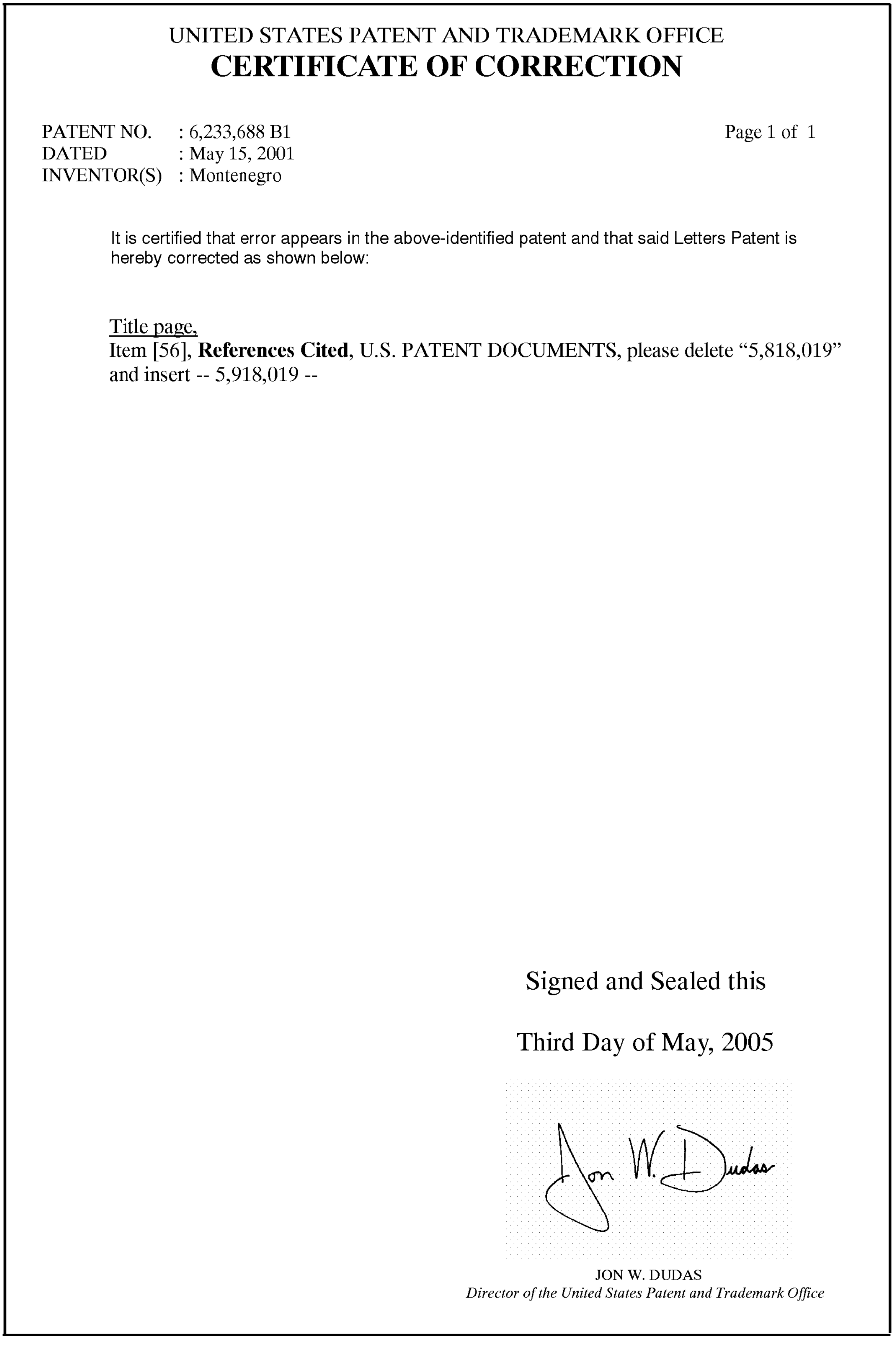
8. A method according to claim **2** further comprising the step of:

performing said specific remote access method, and if said performing is successful, providing data transport resources that extends to said private resource to said client.

9. A computer readable medium comprising:

instructions when executed by a processor cause said processor to provide remote access firewall traversal, said instructions including a remote access firewall traversal (RAFT) uniform resource locator (URL, said RAFT URL indicating a mode of remote access through a firewall.

* * * * *



UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO.	: 6,233,688 B1	Page 1 of 1
DATED	: May 15, 2001	
INVENTOR(S)	: Montenegro	

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title page,
Item [56], **References Cited**, U.S. PATENT DOCUMENTS, please delete "5,818,019" and insert -- 5,918,019 --

Signed and Sealed this

Third Day of May, 2005

Jon W. Dudas

JON W. DUDAS
Director of the United States Patent and Trademark Office

US006484206B2

(12) **United States Patent**
Crump et al.

(10) **Patent No.:** **US 6,484,206 B2**
(45) **Date of Patent:** ***Nov. 19, 2002**

(54) **EFFICIENT RECOVERY OF MULTIPLE CONNECTIONS IN A COMMUNICATION NETWORK**

(75) Inventors: **Richard Crump**, Boston, MA (US); **Todd Short**, Maynard, MA (US)

(73) Assignee: **Nortel Networks Limited**, St. Laurent (CA)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/167,746**

(22) Filed: **Oct. 7, 1998**

(65) **Prior Publication Data**

US 2002/0078208 A1 Jun. 20, 2002

(51) **Int. Cl.**^{7} **G06F 15/16**

(52) **U.S. Cl.** **709/227**; 709/223; 709/230; 709/237; 709/250; 370/401; 370/466; 370/467; 714/4

(58) **Field of Search** 709/200, 203, 709/217–221, 223–224, 227–228, 230–237, 238–239, 242–246, 249–250; 714/2–4, 15, 19–20, 712; 710/11–14; 370/400–401, 466–467

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,021,949	A		6/1991	Morten et al.	709/231
5,023,873	A	*	6/1991	Stevenson et al.	714/4
5,182,748	A		1/1993	Sakata et al.	370/466
5,636,212	A		6/1997	Ikeda	370/233
5,652,908	A	*	7/1997	Douglas et al.	709/228
5,793,771	A		8/1998	Darland et al.	370/467
5,802,258	A	*	9/1998	Chen	709/227
5,838,989	A	*	11/1998	Hutchison et al.	710/11
5,856,981	A	*	1/1999	Voelker	714/712
5,918,017	A	*	6/1999	Attanasio et al.	709/224
5,918,022	A		6/1999	Batz et al.	709/236
5,933,422	A	*	8/1999	Kusano et al.	709/239
5,943,481	A		8/1999	Wakeland	709/230
6,044,407	A		3/2000	Jones et al.	709/246
6,055,224	A		4/2000	King	370/217
6,098,116	A	*	8/2000	Nixon et al.	709/220
6,131,125	A	*	10/2000	Rostoker et al.	709/227
6,134,640	A		10/2000	Unno et al.	711/171
6,192,409	B1		2/2001	Kim	709/230
6,226,676	B1		5/2001	Crump et al.	709/227
6,311,222	B1	*	10/2001	Crump et al.	709/230

OTHER PUBLICATIONS

Stevens, W. Richard, "TCP Establishment and Termination," *TCP/IP Illustrated, vol. 1*, Addison–Wesley Longman, Inc., 1994, ISBN 0–201–63346–9 (v.1).

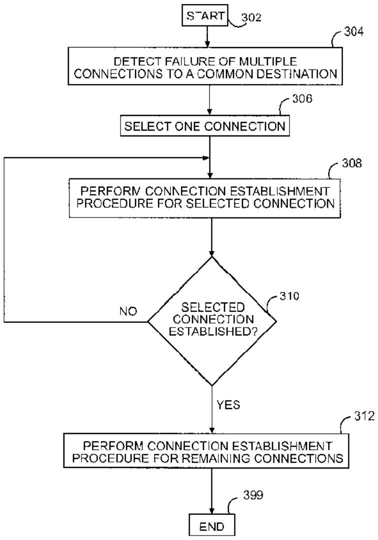
* cited by examiner

Primary Examiner—Bharat Barot
(74) *Attorney, Agent, or Firm*—Withrow & Terranova, P.L.L.C.

(57) **ABSTRACT**

An efficient technique for recovering multiple connections detects a connection failure affecting multiple clients, selects one of the number of affected clients, repeatedly attempts to reestablish the failed connection between the selected client and the server until the connection is successfully re-established, and then re-establishes the connections for the remaining clients.

23 Claims, 8 Drawing Sheets


START
302
304
DETECT FAILURE OF MULTIPLE CONNECTIONS TO A COMMON DESTINATION
306
SELECT ONE CONNECTION
308
PERFORM CONNECTION ESTABLISHMENT PROCEDURE FOR SELECTED CONNECTION
310
SELECTED CONNECTION ESTABLISHED?
NO
YES
312
PERFORM CONNECTION ESTABLISHMENT PROCEDURE FOR REMAINING CONNECTIONS
399
END

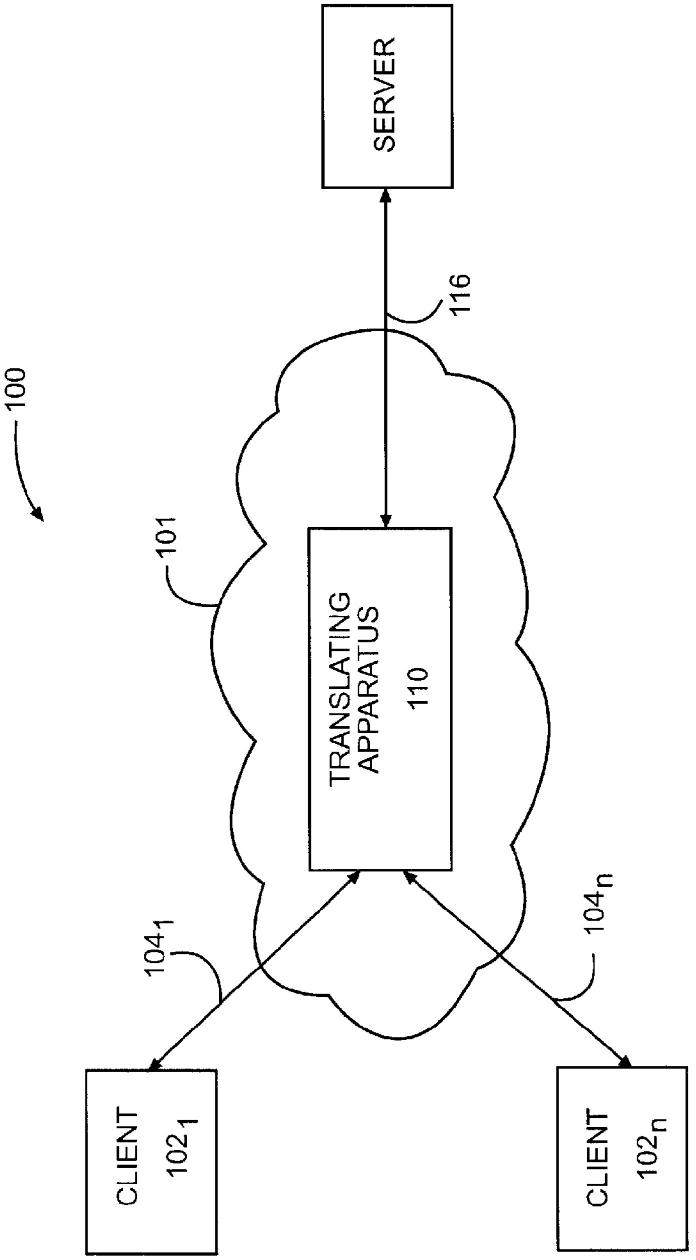

100
101
SERVER
116
TRANSLATING APPARATUS 110
104_1
104_n
CLIENT 102_1
CLIENT 102_n

FIG. 1

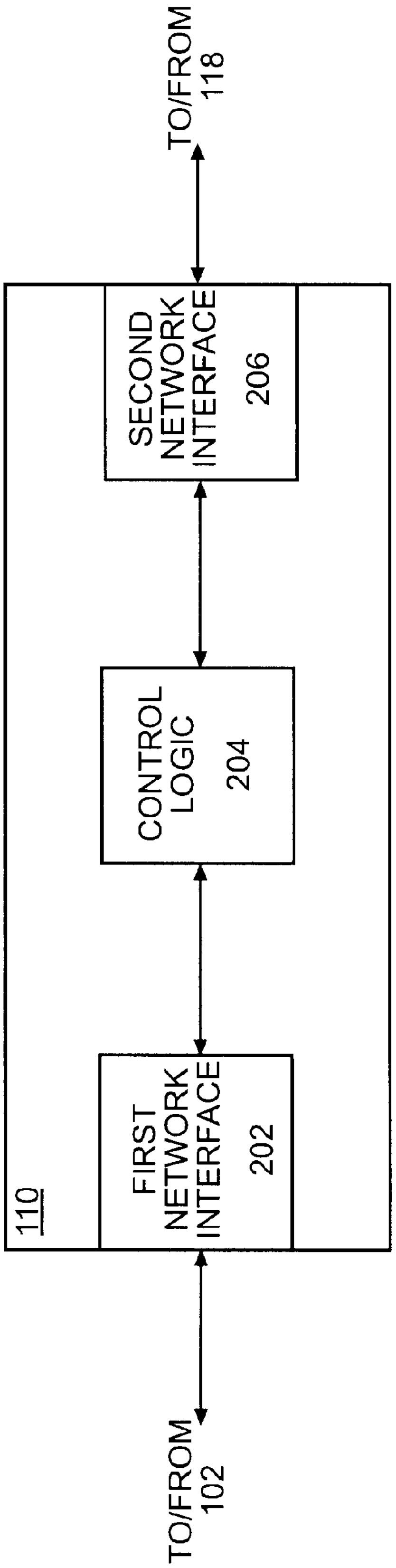

TO/FROM
102
110
FIRST
NETWORK
INTERFACE
202
CONTROL
LOGIC
204
SECOND
NETWORK
INTERFACE
206
TO/FROM
118

FIG. 2

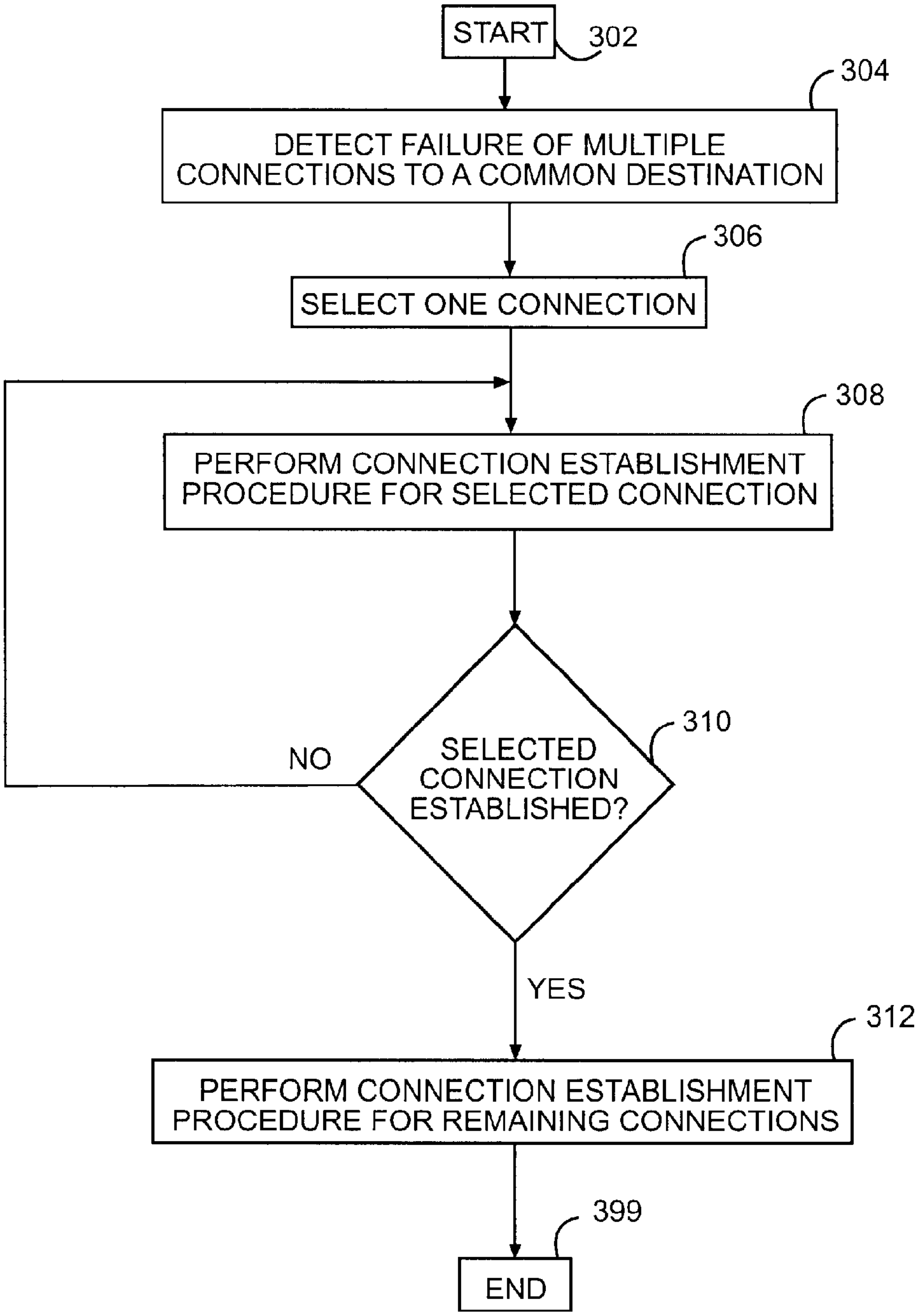

START
302
304
DETECT FAILURE OF MULTIPLE CONNECTIONS TO A COMMON DESTINATION
306
SELECT ONE CONNECTION
308
PERFORM CONNECTION ESTABLISHMENT PROCEDURE FOR SELECTED CONNECTION
310
SELECTED CONNECTION ESTABLISHED?
NO
YES
312
PERFORM CONNECTION ESTABLISHMENT PROCEDURE FOR REMAINING CONNECTIONS
399
END

FIG. 3

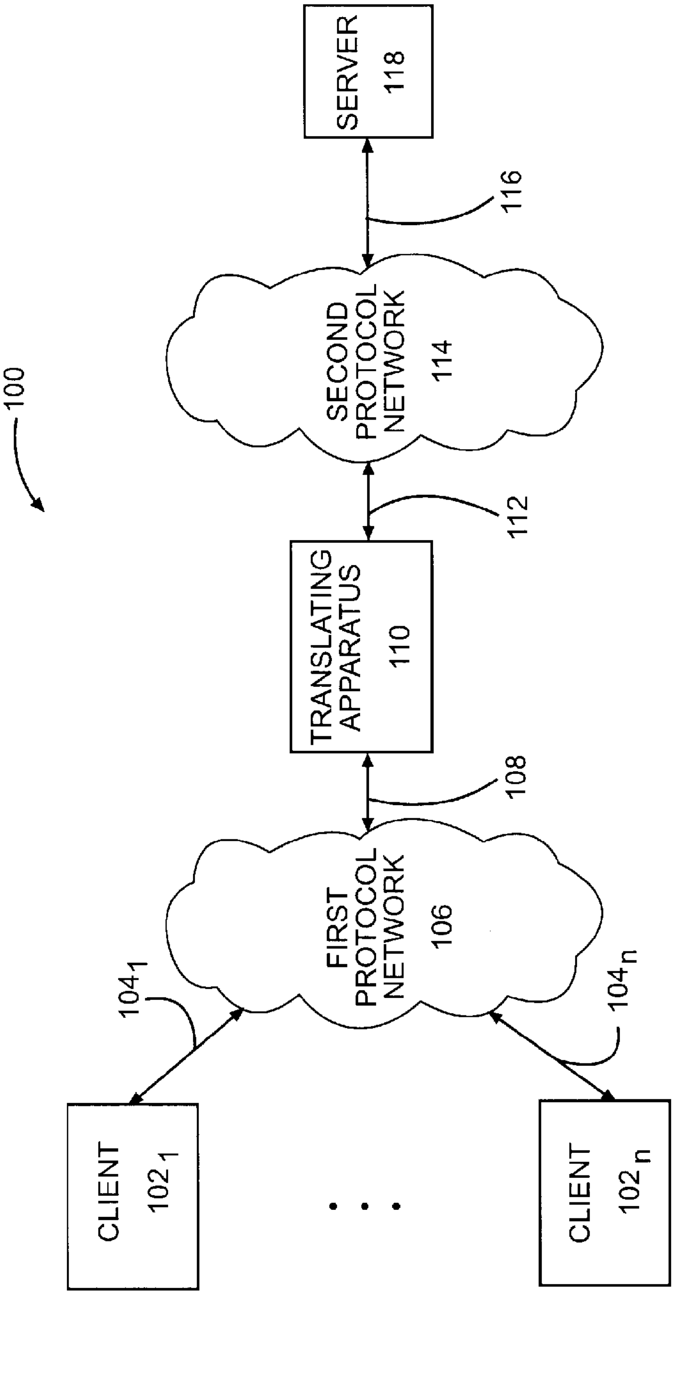

100
SERVER 118
116
SECOND PROTOCOL NETWORK 114
112
TRANSLATING APPARATUS 110
108
FIRST PROTOCOL NETWORK 106
104₁
104ₙ
CLIENT 102₁
. . .
CLIENT 102ₙ

FIG. 4A

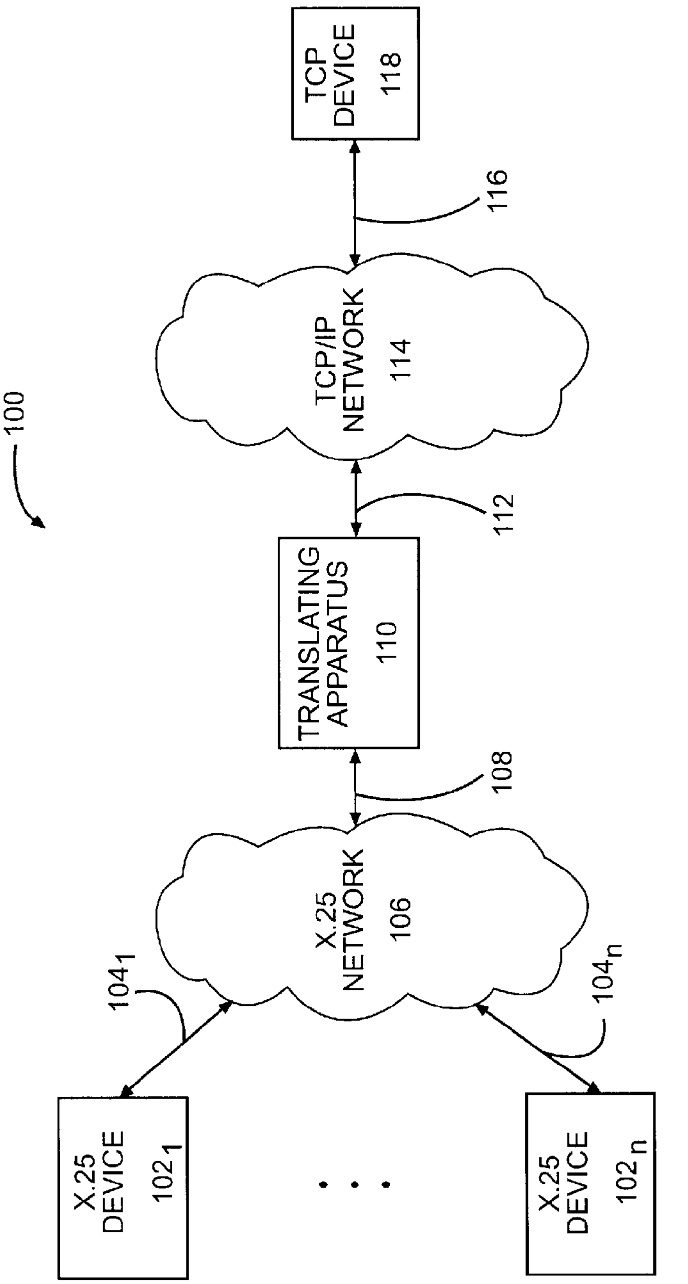

100
TCP DEVICE 118
116
TCP/IP NETWORK 114
112
TRANSLATING APPARATUS 110
108
X.25 NETWORK 106
104_1
104_n
X.25 DEVICE 102_1
. . .
X.25 DEVICE 102_n

FIG. 4B

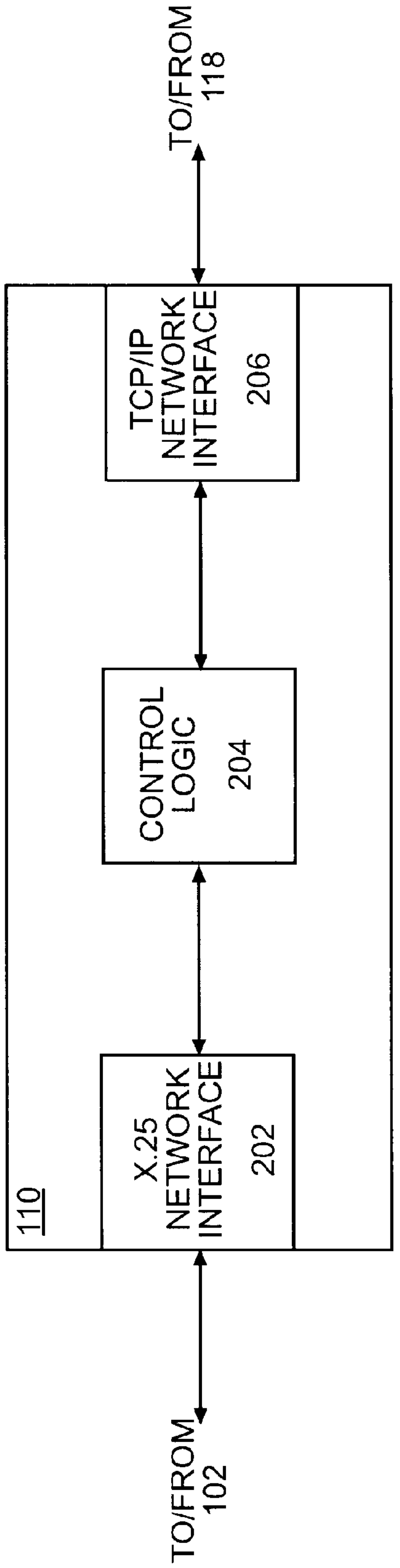

TO/FROM
118
TCP/IP
NETWORK
INTERFACE
206
CONTROL
LOGIC
204
110
X.25
NETWORK
INTERFACE
202
TO/FROM
102

FIG. 5

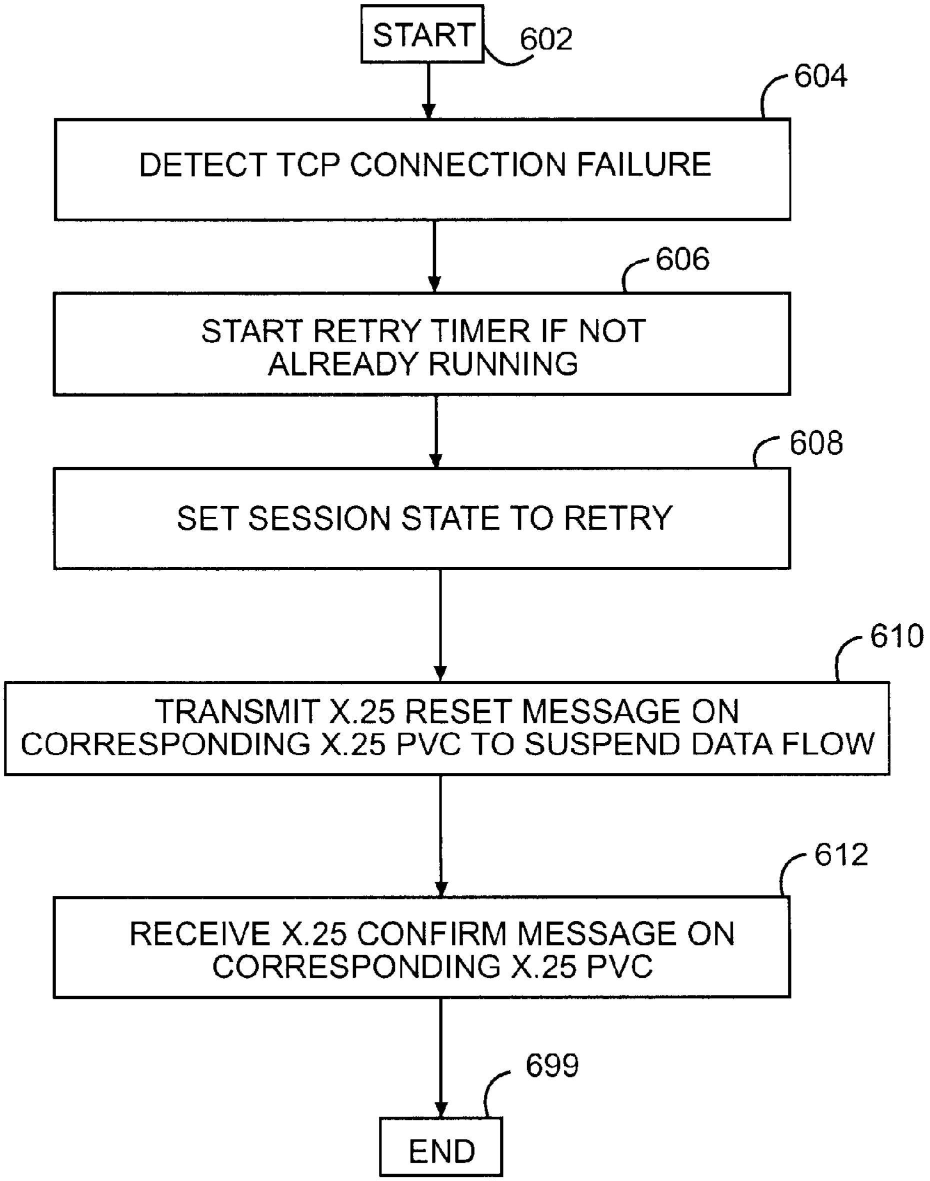

START
602
604
DETECT TCP CONNECTION FAILURE
606
START RETRY TIMER IF NOT ALREADY RUNNING
608
SET SESSION STATE TO RETRY
610
TRANSMIT X.25 RESET MESSAGE ON CORRESPONDING X.25 PVC TO SUSPEND DATA FLOW
612
RECEIVE X.25 CONFIRM MESSAGE ON CORRESPONDING X.25 PVC
699
END

FIG. 6

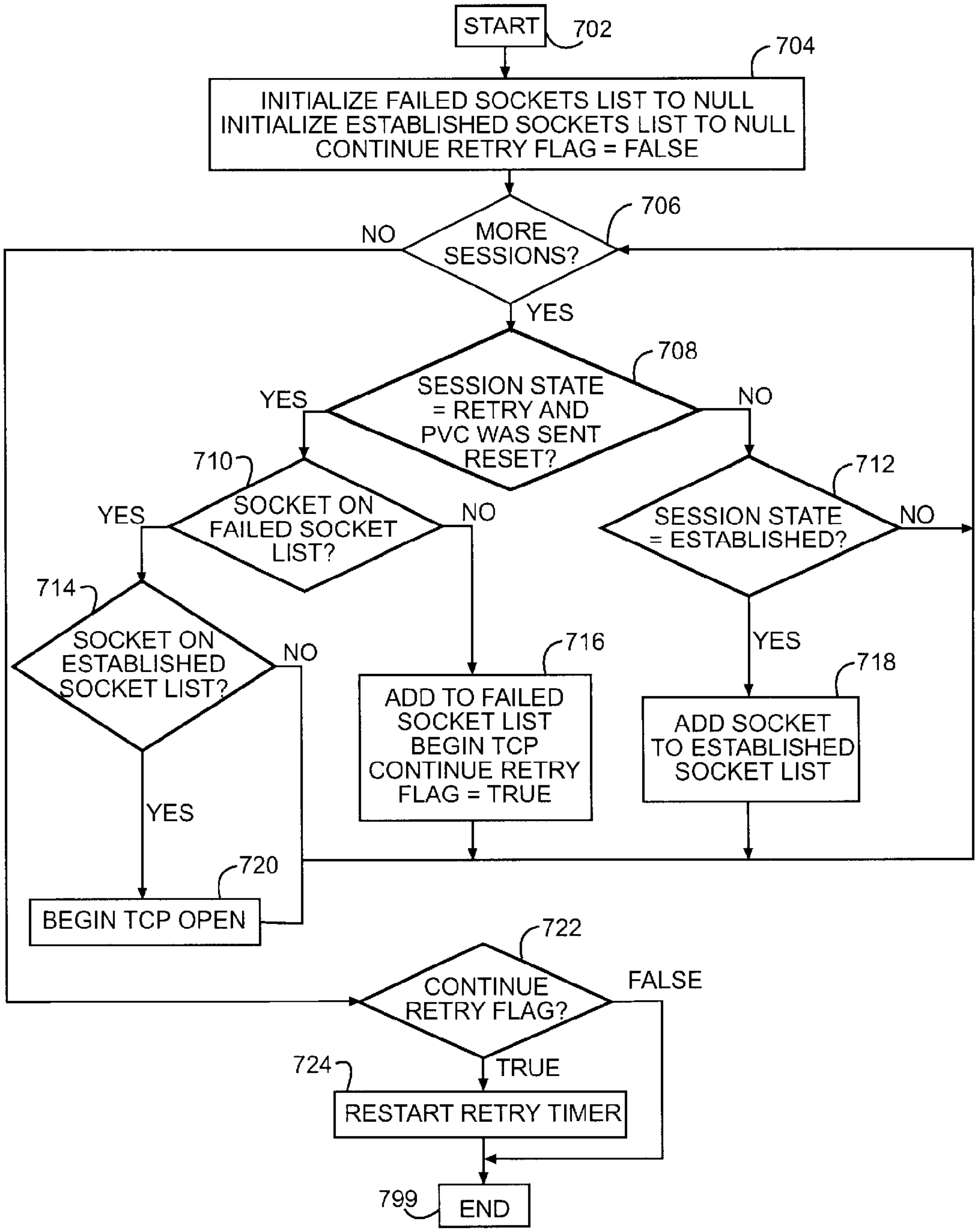

START
702
704
INITIALIZE FAILED SOCKETS LIST TO NULL
INITIALIZE ESTABLISHED SOCKETS LIST TO NULL
CONTINUE RETRY FLAG = FALSE
706
MORE SESSIONS?
NO
YES
708
SESSION STATE = RETRY AND PVC WAS SENT RESET?
YES
NO
710
SOCKET ON FAILED SOCKET LIST?
YES
NO
712
SESSION STATE = ESTABLISHED?
NO
YES
714
SOCKET ON ESTABLISHED SOCKET LIST?
NO
YES
716
ADD TO FAILED SOCKET LIST
BEGIN TCP
CONTINUE RETRY FLAG = TRUE
718
ADD SOCKET TO ESTABLISHED SOCKET LIST
720
BEGIN TCP OPEN
722
CONTINUE RETRY FLAG?
FALSE
TRUE
724
RESTART RETRY TIMER
799
END

FIG. 7

EFFICIENT RECOVERY OF MULTIPLE CONNECTIONS IN A COMMUNICATION NETWORK

CROSS-REFERENCES TO RELATED APPLICATIONS

This patent application may be related to the following commonly-owned United States patent applications:

U.S. patent application Ser. No. 09/167,916 entitled CONNECTION ESTABLISHMENT AND TERMINATION IN A MIXED PROTOCOL NETWORK, filed on even date now U.S. Pat. No. 6,226,676 and hereby incorporated by reference in its entirety;

U.S. patent application Ser. No. 09/167,950 entitled ERROR RECOVERY IN A MIXED PROTOCOL NETWORK, filed on even date herewith, still pending and hereby incorporated by reference in its entirety;

U.S. patent application Ser. No. 09/167,839 entitled ESTABLISHING AND TERMINATING CONNECTIONS IN A MIXED PROTOCOL NETWORK, filed on even date herewith, now U.S. Pat. No. 6,320,874 and hereby incorporated by reference in its entirety;

U.S. patent application Ser. No. 09/167,792 entitled SYSTEM FOR TRANSLATING A MESSAGE FROM A FIRST TRANSMISSION PROTOCOL TO A SECOND TRANSMISSION PROTOCOL, filed on even date herewith, still pending and hereby incorporated by reference in its entirety; and

U.S. patent application Ser. No. 09/167,811 entitled TRANSLATOR MEMORY MANAGEMENT SYSTEM, filed on even date herewith, now U.S. Pat. No. 6,311,222 and hereby incorporated by reference in its entirety.

FIELD OF THE INVENTION

The present invention relates generally to data communication networks, and, more particularly, to recovery of multiple connections in a communication network.

BACKGROUND OF THE INVENTION

In today's information age, data communication networks are becoming ever more pervasive as an ever-increasing number of communication consumers require access to on-line computer resources. To that end, many data communication networks are evolving to meet the needs of these communication consumers. In order to support a large number of users. it is important for the protocols used in the data communication networks to be efficient.

A common network configuration (referred to hereinafter as the "client-server model") includes a number of client devices that communicate with a common server over the communication network. In this client-server model, each client establishes a connection to the server over the communication network. Thus, the server represents a common destination for all of its connected clients.

In the client-server model, it is typical for client-server connections to be established and terminated dynamically. For example, it is common for existing clients to disconnect from the server and for new clients to connect to the server. When connections are established and terminated using a prescribed protocol, there is generally no disruption to other existing connections.

Under certain circumstances, though, it is possible for multiple clients to become disconnected from the server. When this occurs, it is typical for the affected clients to try to reestablish the connections to the server. When multiple clients attempt to connect to the server simultaneously, the communication network can become inundated with protocol messages sent by the clients. Furthermore, it is likely that the simultaneous attempts to re-establish the connections to the server will either all fail, if the server remains inaccessible, or all succeed, if the server becomes accessible. Therefore, an efficient technique for recovering multiple connections is needed.

SUMMARY OF THE INVENTION

In accordance with one aspect of the invention, a method, translating apparatus, apparatus comprising a computer readable medium, and system includes a translating function for recovering multiple connections in a communication network. The translating function detects a failure affecting a plurality of connections, selects one of said plurality of connections, performs a connection establishment procedure to re-establish the selected connection, and, upon successfully re-establishing the selected connection, performs the connection establishment procedure for the remainder of the plurality of connections.

In a preferred embodiment of the invention, the translating function utilizes timer-driven connection recover logic to periodically re-establish failed connections. The timer is started upon detection of an initial connection failure.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects and advantages of the invention will be appreciated more fully from the following further description thereof with reference to the accompanying drawings wherein:

FIG. 1 shows a client-server communication network in accordance with a preferred embodiment of the present invention;

FIG. 2 is a block diagram of an exemplary translating apparatus in accordance with a preferred embodiment of the present invention;

FIG. 3 is a logic flow diagram showing exemplary connection recovery logic for reestablishing the client connections in accordance with an embodiment of the present invention:

FIG. 4A is a block diagram of an exemplary data communication network in which a translating apparatus is used to allow clients, which communicate with the translating apparatus over a first protocol network, to communicate with a server, which communicates with the translating apparatus over a second protocol network;

FIG. 4B is a block diagram of a preferred embodiment of the exemplary data communication network in which the translating apparatus is used to allow X.25 devices, which communicate with the translating apparatus over an X.25 network, to communicate with a TCP device, which communicates with the translating apparatus over a TCP/IP network;

FIG. 5 is a block diagram of a preferred translating device for use in a mixed X.25 /TCP communication network;

FIG. 6 is a logic flow diagram for initiating a restart timer upon detection of a connection failure in accordance with a preferred embodiment of the present invention; and

FIG. 7 is a logic flow diagram for connection recovery logic in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

As discussed above, an efficient technique for recovering multiple connections is needed. The present invention provides such an efficient technique for recovering multiple connections by selecting one of the affected clients, repeatedly attempting to re-establish the failed connection between the selected client and the server until the connection is successfully re-established, and then re-establishing the connections for the remaining clients. In this way, only one client attempts to connect to the server while the server is inaccessible, and the remaining clients only attempt to re-establish the connection after the server becomes accessible.

FIG. **1** shows a number of clients **102_1** through **102_n** (referred to collectively as "clients **102**") in communication with a server **118** over a communication network **101**. The clients **102** are coupled to the communication network **101** by way of links **104_1** through **104_n**. The server **118** is coupled to the communication network by way of link **116**.

Continuing to refer to FIG. **1**, the communication network **101** includes a translating apparatus **110**. The translating apparatus **110** is in communication with the clients **102** and the server **118**. The translating apparatus **110** facilitates communication between the clients **102** and the server **118** by establishing connections between the clients **102** and the server **118**, terminating connections between the clients **102** and the server **118**, recovering from connection failures between the clients **102** and the server **118**, and transferring application data between the clients **102** and the server **118**.

FIG. **2** is a block diagram of an exemplary translating apparatus **110** in accordance with a preferred embodiment of the present invention. The translating apparatus **110** includes a first network interface **202** that is couplable to the clients **102** and a second network interface **206** that is couplable to the server **118**.

Continuing to refer to FIG. **2**, the translating apparatus **110** further includes control logic **204**. The control logic **204** performs a translation function to allow the clients **102** to communicate with the server **118**. The control logic **204** is operably coupled to the first network interface **202**, through which the control logic **204** communicates with the clients **102**. The control logic **204** is also operably coupled to the second network interface **206**, through which the control logic **204** communicates with the server **118**.

The translating apparatus **110** acts as a proxy in order to enable communication between the clients **102** and the server **118**. Specifically, the translating apparatus **110** functions as a server when communicating with the clients **102** and functions as a client when communicating with the server **118**. The control logic **204** controls connection establishment, connection termination, recovery from failed connections, and data transfers on behalf of the clients **102** and the server **118**.

Because the translating apparatus **110** is situated such that the connections between the clients **102** and the server **118** go through the translating apparatus **11(0**, the control logic **204** is able to monitor the status of those connections. Therefore, when a connection fails unexpectedly, the control logic **204** takes steps of re-establish the failed connection.

When communication between the translating apparatus **110** and the server **118** is lost, all client connections to the server **118** are lost. In this situation, the translating apparatus **110**, and particularly the control logic **204**, attempts to re-establish all of the connections between the clients **102** and the server **118**.

In the prior art, each client connection would be handled separately by the control logic **204**, and therefore the control logic **204** would attempt to simultaneously re-establish connections for all of the clients **102**. In this situation, it is likely that the simultaneous attempts to re-establish the connections to the server would either all fail, if the server remains inaccessible, or all succeed, if the server becomes accessible.

In a preferred technique for re-establishing the client connections, the translating apparatus **110** detects a failure affecting multiple client connections. Then, instead of attempting to re-establish all of the client connections simultaneously, the translating apparatus **110** selects one of the failed client connections and attempts to re-establish only the selected client connection. Once the selected client connection is re-established, the remaining client connections are re-established.

Thus, the translating apparatus **110**, and particularly the control logic **204**, includes connection recovery logic for re-establishing the client connections. More specifically, the control logic **204** includes logic for performing a set of steps as shown in FIG. **3**. After beginning in step **302**, and upon detecting a failure affecting multiple connections to the server **118** in step **304**, the control logic **204** selects one connection from among the plurality of connections, in step **306**, and performs a connection establishment procedure for the selected connection, in step **308**. The control logic **204** then determines whether or-not the selected connection was successfully re-established, in step **310**. If the selected connection failed to be established (NO in step **310**), then the control logic **204** repeats the connection establishment procedure of step **308**. However, if the selected connection was successfully established (YES in step **310**), then the control logic **204** proceeds to perform the connection establishment procedure for the remaining client connections, in step **312**, and terminates in step **399**.

FIG. **4**A shows an exemplary communication network **100** in accordance with a preferred embodiment of the present invention. In the exemplary communication network **100** as shown in FIG. **4**A, the clients **102** communicate with the translating apparatus **110** over a first protocol network **106** using a first communication protocol, and the translating apparatus **110** communicates with the server **118** over a second protocol network **114** using a second communication protocol. More specifically, the clients **102** are coupled to the first protocol network **106** by way of a links **104_1** through **104_n**. The translating apparatus **110** is also coupled to the first protocol network **106** by way of a link **108**. The clients **102** communicate with the translating apparatus **110** over the first protocol network **106** using the first communication protocol, and, in particular, over connections that are established between the clients **102** and the translating apparatus **110**.

Continuing to refer to FIG. **4**A, the server **118** is coupled to a second protocol network **114** by way of a link **116**. The translating apparatus **110** is also coupled to the second protocol network **114** by way of a link **112**. The server **118** communicates with the translating apparatus **110** over the second protocol network **114** using the second communication protocol, and in particular, over a connection that is established between the server **118** and the translating apparatus **110**.

In accordance with a preferred embodiment of the present invention, the first communication protocol is a protocol known as the X.25 protocol. The X.25 protocol defines the physical, link, and network layer protocols (layers one, two, and three) of the International Standards Organization (ISO) seven-layer protocol model. In a communication network that utilizes the X.25 protocol (referred to herein as an "X.25 network"), two devices (referred to herein as an "X.25

device" or "X.25 devices") exchange X.25 network layer messages (referred to in X.25 as "packets") over a virtual circuit that is established across the X.25 network. One type of virtual circuit commonly used in the X.25 network is a permanent virtual circuit or PVC. A PVC is a virtual circuit that is set up automatically within the X.25 network and remains active as long as the X.25 network is operative (as opposed to a switched virtual circuit or SVC, which is set up only when explicitly requested by an X.25 device). Typical X.25 networks support multiple virtual circuits, both permanent and switched.

In accordance with a preferred embodiment of the present invention, the second communication protocol is a protocol known as the Transmission Control Protocol (referred to hereinafter as "TCP"). TCP is a connection-oriented transport layer protocol that is generally used in conjunction with a connectionless network layer protocol known as the Internet Protocol or IP. In a communication network that utilizes the TCP protocol (referred to herein as a "TCP/IP network"), two devices (referred to herein as a "TCP device" or "TCP devices") exchange TCP messages (referred to in TCP as "segments") over a TCP connection that is established across the TCP/IP network. In order to set up the TCP connection within the TCP/IP network, two TCP devices exchange specially formatted messages that include, among other things, an IP address identifying the destination TCP device and a TCP port number identifying one of a number of applications supported by the destination TCP device. The combination of IP address and TCP port number is referred to hereinafter as a "socket." Because the TCP connection is set up only when explicitly requested by a TCP device, the TCP connection is considered to be a switched connection as opposed to a permanent connection.

FIG. 4B shows a preferred embodiment of the data communication network **100** in which the first communication protocol is the X.25 protocol and the second communication protocol is the TCP protocol. Thus, in accordance with a preferred embodiment of the present invention as shown in FIG. 4B, the X.25 devices **102** communicate with the translating apparatus **110** across an X.25 network **106** using the X.25 protocol, and the TCP device **118** communicates with the translating apparatus **110** across a TCP/IP network **114** using the TCP protocol. The preferred translating apparatus **110**, as shown in FIG. **5**, therefore includes an X.25 network interface **202** for interfacing with the X.25 network **106** and a TCP/IP network interface **206** for interfacing with the TCP/IP network **114**.

In order for the X.25 devices **102** to communicate with the TCP device **118** within the data communication network **100**, it is necessary for end-to-end connections to be established between the X.25 devices **102** and the TCP device **118**. In order for an end-to-end connection to exist between an X.25 device **102** and the TCP device **118**, there must be both an active X.25 connection between the X.25 device **102** and the translating apparatus **110** and an active TCP connection between the translating apparatus **110** and a particular socket in the TCP device **118** (referred to hereinafter as the "destination socket"). In accordance with a preferred embodiment of the present invention, the X.25 devices **102** communicate with the translating apparatus **110** over dedicated X.25 PVCs, and therefore active X.25 connections exist between the X.25 devices **102** and the translating apparatus **110** as long as the X.25 network **106** is operative. On the other hand, because the TCP device **118** communicates with the translating apparatus **110** over a switched TCP connection, the TCP connection between the translating apparatus **110** and a the destination socket is established dynamically.

Furthermore, in order for the translating device **10** to provide an end-to-end connection between the X.25 devices **102** and the TCP device **118**, the translating apparatus **110**, and particularly the control logic **204**, maintains a map associating the X.25 PVCs with their corresponding destination sockets. In accordance with a preferred embodiment of the present invention, one X.25 PVC maps to one destination socket, and multiple X.25 PVCs can map to the same destination socket as is typically the case when multiple X.25 client devices **102** are communicating with the same TCP server device **118**. Application data received from the X.25 devices **102** over the X.25 PVCs is transmitted by the translating apparatus **110**, and particularly by the control logic **204**, to the TCP device **118** over the corresponding TCP connections. Likewise, application data received from the TCP device **118**. over the TCP connections is transmitted by the translating apparatus **110**, and particularly by the control logic **204**, to the X.25 devices **102** over the corresponding X.25 PVCs. Exemplary embodiments of control logic **204** for transferring application data between the X.25 device **102** and the TCP device **118** are described in the related U.S. patent applications Ser. No. 09/167,702 still pending and Ser. No. 09/167,811 now U.S. Pat. No. 6,311, 22.

When communication between the X.25 devices **102** and the TCP device **118** is completed, it is often desirable for the end-to-end connections between the X.25 devices **102** and the TCP device **118** to be terminated. Because the X.25 devices **102** communicate with the translating apparatus **110** over dedicated X.25 PVCs, the X.25 connections between the X.25 devices **102** and the translating apparatus **110** cannot be terminated. On the other hand, because the TCP device **118** communicates with the translating apparatus **110** over a switched TCP connection, the TCP connection between the translating apparatus **110** and a particular socket in the TCP device **118** is terminated dynamically.

In the data communication network **100**, connection establishment and termination may be initiated by either the X.25 devices **102** or the TCP device **118**. The related U.S. patent application Ser. No. 09/167,916 now U.S. Pat. No. 6,226,676 describes a first exemplary embodiment in which the X.25 devices **102** initiate connection establishment and termination. The related U.S. patent application Ser. No. 09/167,839 now U.S. Pat No. 6,320,874 describes a second exemplary embodiment in which the TCP device **118** initiates connection establishment and termination.

Under certain circumstances, an attempt to establish a TCP connection between the translating apparatus **110** and the TCP device **118** may fail. For example in the connection, establishment mechanism described in the related U.S. patent application Ser. No. 09/167,916 now U.S. Pat. No. 6,226,676 in which the translating apparatus **110** performs a TCP open procedure with the TCP device **118** upon receiving an X.25 data packet from the X.25 device **102**, it is conceivable that message transfers between the translating apparatus **110** and the TCP device **118** will be lost, resulting in a failure of the translating apparatus **110** to establish the TCP connection with the TCP device **118**. If this occurs, it is desirable for the translating apparatus **110** to suspend data flow between the X.25 device **102** and the translating apparatus **118**, repeat the TCP connection establishment mechanism until a TCP connection is established between the translating apparatus **110** and the TCP device **118**, and then enable data flow between the X.25 device **102** and the translating apparatus **118**.

Under other circumstances, an existing TCP connection between the translating apparatus **110** and the TCP device

118 may fail or be explicitly terminated. For example, a network error may cause the existing TCP connection to be dropped, or the existing TCP connection may be explicitly terminated using the connection termination mechanism described in the related U.S. patent application Ser. No. 09/167,859 now U.S. Pat. No. 6,320,874. If an existing TCP connection is lost or terminated, it is often desirable for the translating apparatus **110** to suspend data flow between the X.25 device **102** and the translating apparatus **118**, repeat the TCP connection establishment mechanism until a TCP connection is established between the translating apparatus **110** and the TCP device **118** or other TCP device, and then enable data flow between the X.25 device **102** and the translating apparatus

The related U.S. patent application Ser. No. 09/167,950 still depending describes a general error recovery technique in which the translating apparatus **110** suspends data flow with the X.25 device upon detecting a TCP connection failure and enables data flow with the X.25 device upon re-establishing the TCP connection. The affected TCP connection(s) are re-established while data flow is suspended on the affected X.25 PVC(s).

A preferred embodiment of the present invention provides an efficient technique for recovering multiple failed connections to the same destination socket. Specifically, upon detecting a TCP connection failure to a particular destination socket, a session state indicator associated with the destination socket (indicating whether the connection to the destination socket is in an ESTABLISHED state or a RETRY state) is set to the RETRY state, and a retry timer is started (if the retry timer is not already running).

Thus, the translating apparatus **110**, and particularly the control logic **204**, includes logic for starting the retry timer. More specifically, the control logic **204** includes logic for performing a set of steps as shown in FIG. **6**. After beginning at step **602**, and upon detecting a TCP connection failure in step **604**, the control logic **204** starts the retry timer if the retry timer is not already running, in step **606**. In accordance with a preferred embodiment of the present invention, the retry timer is set to run after fifteen (15) seconds. The control logic **204** then sets the session state indicator associated with the destination socket equal to the RETRY state in step **608**. When used in conjunction with the error recovery mechanism described in the related U.S. patent application Ser. No. 09/167,950 still pending, the control logic **204** further transmits an X.25 reset message to each affected X.25 PVC, in step **610**, and receives an X.25 confirm message on each of the affected X.25 PVCs, in step **612**. The control logic **204** terminates in step **699**.

When the retry timer expires, the control logic **204** performs connection recovery logic. The connection recovery logic maintains two lists, a failed sockets list and an established sockets list. The connection recovery logic also maintains a retry indicator, referred to hereinafter as the "Continue Retry Flag."

Thus, the translating apparatus **110**, and particularly the control logic **204**, includes connection recovery logic for performing a set of steps as shown in FIG. **7**. After beginning at step **702**, the control logic **204** clears the failed sockets list and the established sockets list (i.e , the lists are initialized to NULL), and sets the Continue Retry Flag is equal to FALSE in step **704**. The control logic **204** then cycles through each session (i.e., connection) in a session list as follows.

If there is a session to be processed (YES in step **706**), the control logic **204** checks the status of the session, in step **708**. Specifically, if the session state indicator associated with the session is equal to the RETRY state and the corresponding X.25 PVC was sent a reset (YES in step **708**), then the control logic **204** checks whether or not the socket associated with the session is on the failed sockets list, in step **710**. If the socket is not on the failed sockets list (NO in step **710**), then the control logic **204** adds the socket to the failed sockets list, begins a TCP open procedure to establish a connection to the destination socket, and sets the Continue Retry Flag equal to TRUE, in step **716**, and recycles to step **706** to process the next socket.

If the socket is on the failed sockets list (YES in step **710**), then the control logic **204** checks whether or not the socket is on the established sockets list, in step **714**. If the socket is not on the established sockets list (NO in step **714**), then the control logic **204** recycles to step **706** to process the next socket. If the socket is on the established sockets list (YES in step **714**), then the control logic **204** begins a TCP open procedure to establish a connection to the destination socket, in step **720**, and recycles to step **706** to process the next session.

Returning to step **708**, if the session state indicator is equal to the ESTABLISHED state or the corresponding X.25 PVC was not sent a reset (NO in step **708**), then the control logic **204** checks whether the session state indicator is equal to the ESTABLISHED state, in step **712**. If the session state indicator is not equal to the ESTABLISHED state (NO in step **712**), then the control logic **204** recycles to step **706** to process the next session. If the session state indicator is equal to the ESTABLISHED state (YES in step **712**), then the control logic **204** adds the associated socket to the established sockets list, in step **718**, and recycles to step **706** to process the next session.

When all sessions have been processed (NO in step **706**), the control logic **204** checks the Continue Retry Flag, in step **722**. If the Continue Retry Flag is equal to FALSE in step **722**, then the control logic terminates in step **799**. If the Continue Retry Flag is equal to TRUE in step **722**, then the control logic **204** restarts the retry timer in step **724**, and terminates in step **799**.

Although the various embodiments are described with specific reference to a translating apparatus for enabling communication between an X.25 device and a TCP device, it will be apparent to a skilled artisan that the techniques of the present invention apply more generally to a translating apparatus for enabling communication between two devices supporting different communication protocols. Thus, the translating apparatus enables communication between a first device utilizing a first communication protocol and a second device utilizing a second communication protocol. More specifically, the first device communicates with the translating apparatus using the first communication protocol, and the second device communicates with the translating apparatus using the second communication protocol. The translating apparatus performs the functions necessary to establish an end-to-end connection between the first device and the second device, for example, as described in the related U.S. patent applications Ser. No. 09/167,916 now U.S. Pat. No. 6,226,676 and Ser. No. 09/167,839 now U.S. Pat. No. 6,320,879 terminate an end-to-end connection device, for example, as described in the related U.S. patent applications Ser. No. 09/167,916 now U.S. Pat No. 6,226,672 and Ser. No. 09/167,839 now U.S. Pat. No. 6,320,874; recover from a connection failure, for example, as described herein and in the related U.S. application Ser. No. 09/167,950 still pending; and exchange application data, for example, as described in the related U.S. patent applications Ser. No.

09/167,792 now U.S. Pat. No. 6,226,075 and Ser. No. 09/167,811 now U.S. Pat. No. 6,311,222.

In a preferred embodiment of the present invention, the control logic **204** is implemented as a set of program instructions that are stored in a computer readable memory within the translating apparatus **110** and executed on a microprocessor within the translating apparatus **110**. However, it will be apparent to a skilled artisan that all logic described herein can be embodied using discrete components, integrated circuitry, programmable logic used in conjunction with a programmable logic device such as a Field Programmable Gate Array (FPGA) or microprocessor, or any other means including any combination thereof. Programmable logic can be fixed temporarily or permanently in a tangible medium such as a read-only memory chip, a computer memory, a disk, or other storage medium. Programmable logic can also be fixed in a computer data signal embodied in a carrier wave, allowing the programmable logic to be transmitted over an interface such as a computer bus or communication network. All such embodiments are intended to fall within the scope of the present invention.

The present invention may be embodied in other specific forms without departing from the essence or essential characteristics. The described embodiments are to considered in all respects only as illustrative and not restrictive.

We claim:

1. A method for recovering multiple connections in a communication network, the method comprising the steps of:

initially establishing a plurality of connections according to a first protocol to a translating apparatus;

establishing a second plurality of connections from the translating apparatus to a second apparatus with a second protocol, said second protocol different than said first protocol;

translating between said first protocol and said second protocol with said translating apparatus;

detecting a failure affecting said second plurality of connections;

selecting one of said second plurality of connections;

performing a connection establishment procedure to re-establish the selected connection; and

upon successfully re-establishing the selected connection, period the connection establishment procedure for the remainder of the second plurality of connections.

2. The method of claim **1**, wherein the second plurality of connections are Transmission Control Protocol (TCP) connections.

3. The method of claim **2**, wherein the detecting the failure affecting the second plurality of connections comprises:

detecting a TCP connection failure to a destination socket common to the second plurality of connections.

4. The method of claim **2**, wherein performing the connection establishment procedure to reestablish the selected connection comprises:

performing a first TCP open procedure for the selected connection.

5. The method of claim **4**, wherein performing the connection establishing procedure for the remainder of the plurality of connections upon successfully re-establishing the selected connection comprises:

determining that the selected connection has been established through the first TCP open procedure; and

performing a second TCP open procedure for another connection.

6. An apparatus comprising:

a first plurality of connections according to a first protocol;

a second plurality of connections according to a second protocol, said second protocol different from said first protocol;

translation logic operably coupled to said first and second pluralities of connections for translating between said first and second protocols;

failure detection logic operably coupled to detect a failure affecting said second plurality of connections;

selection logic responsive to the failure detection logic and operably coupled to select one of said second plurality of connections;

first connection re-establishment logic responsive to the selection logic and operably coupled to re-establish the selected connection; and

second connection re-establishment logic responsive to the first connection re-establishment logic and operably coupled to re-establish the remaining connections upon successful re-establishment of the selected connection.

7. The apparatus of claim **6**, wherein the second plurality of connections are Transmission Control Protocol (TCP) connections.

8. The apparatus of claim **7**, wherein the failure detection logic is operably coupled to detect a TCP connection failure to a destination socket common to the second plurality of connections.

9. The apparatus of claim **7**, wherein the first connection re-establishment logic is operably coupled to perform a first TCP open procedure for the selected connection.

10. The apparatus of claim **9**, wherein the second connection re-establishment logic is operably coupled to determine that the selected connection has been established by the first connection re-establishment logic and perform a second TCP open procedure for another connection.

11. A computer program for controlling a computer system, the computer program comprising:

first connection establishment logic programmed to establish a first plurality of connections according to a first protocol;

second connection establishment logic programmed to establish a second plurality of connections according to a second protocol, said first protocol different than said second protocol;

translation logic for translating between said first and second protocols and operably associated with said first and second pluralities of connections;

failure detection logic programmed to detect a failure affecting the second plurality of connections,

selection logic responsive to the failure detection logic and programmed to select one of said second plurality of connections;

first connection re-establishment logic responsive to the selection logic and programmed to re-establish the selected connection; and

second connection re-establishment logic responsive to the first connection re-establishment logic and programmed to re-establish the remaining connections upon successful re-establishment of the selected connection.

12. The computer program of claim **11**, wherein the second plurality of connections are Transmission Control Protocol (TCP) connections.

13. The computer program of claim **12**, wherein the failure detection logic is programmed to detect a TCP connection failure to a destination socket common to the second plurality of connections.

14. The computer program of claim **12**, wherein the first connection re-establishment logic is programmed to perform a first TCP open procedure for the selected connection.

15. The computer program of claim **14**, wherein the second connection re-establishment logic is programmed to determine that the selected connection has been established by the first connection re-establishment logic and perform a second TCP open procedure for another connection.

16. The computer program of claim **11** embodied in a computer readable medium.

17. The computer program of claim **11** embodied in a data signal.

18. A communication system comprising a plurality of clients in communication with a server through a translating apparatus, wherein the translating apparatus is operably coupled to establish a plurality of connections between the plurality of clients and the server, wherein the plurality of connections comprises a first plurality of connections using a first protocol and a second plurality of connections using a second protocol, said translating apparatus further adapted to:

translate between said first and second protocols,

detect a failure affecting the second plurality of connections;

select one of said plurality of connections;

re-establish the selected connection, and

re-establish the remaining connections after re-establishing the selected connection.

19. The communication system of claim **18**, wherein the second plurality of connections are Transmission Control Protocol (TCP) connections.

20. The communication system of claim **19**, wherein the translating apparatus is operably coupled to detect a TCP connection failure to a destination socket common to the plurality of connections.

21. The communication system of claim **19**, wherein the translating apparatus is operably coupled to perform a first TCP open procedure for the selected connection in order to re-establish the selected connection.

22. The communication system of claim **21**, wherein the translating apparatus is operably coupled to determine that the selected connection has been established by the first connection re-establishment logic and perform a second TCP open procedure for another connection.

23. The communication system of claim **18**, wherein the clients are X.25 clients, and wherein the server is a Transmission Control Protocol (TCP) server.

* * * * *

US006854063B1

(12) **United States Patent**
Qu et al.

(10) **Patent No.:** **US 6,854,063 B1**
(45) **Date of Patent:** ***Feb. 8, 2005**

(54) **METHOD AND APPARATUS FOR OPTIMIZING FIREWALL PROCESSING**

(75) Inventors: **Diheng Qu**, Santa Clara, CA (US); **Kevin Li**, Milpitas, CA (US); **Sami Boutros**, Kanata (CA); **Seren Fan**, Palo Alto, CA (US); **Steve Truong**, Saratoga, CA (US)

(73) Assignee: **Cisco Technology, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/517,961**

(22) Filed: **Mar. 3, 2000**

(51) **Int. Cl.**^{7} **G06F 15/173**; G06F 15/177; H04L 9/00; H04L 12/28; H04L 12/56

(52) **U.S. Cl.** **713/201**; 370/392; 709/238

(58) **Field of Search** 713/201; 370/392; 709/238

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,796,942	A	*	8/1998	Esbensen	713/201
5,842,040	A		11/1998	Hughes et al.	395/831
5,898,830	A	*	4/1999	Wesinger et al.	713/201
6,067,569	A	*	5/2000	Khaki et al.	709/224
6,147,976	A	*	11/2000	Shand et al.	370/254
6,157,955	A	*	12/2000	Narad et al.	709/228
6,170,012	B1	*	1/2001	Coss et al.	709/229
6,182,149	B1	*	1/2001	Nessett et al.	709/247
6,219,706	B1	*	4/2001	Fan et al.	709/225
6,321,259	B1	*	11/2001	Ouellette et al.	709/220
6,496,935	B1	*	12/2002	Fink et al.	713/201
6,680,941	B1	*	1/2004	Schmitz	370/392

FOREIGN PATENT DOCUMENTS

WO	WO97/24841	10/1997	.... H04L/12/46

OTHER PUBLICATIONS

Blake et al., RFC 2475, “An Architecture for Differentiated Services,” 1998.*

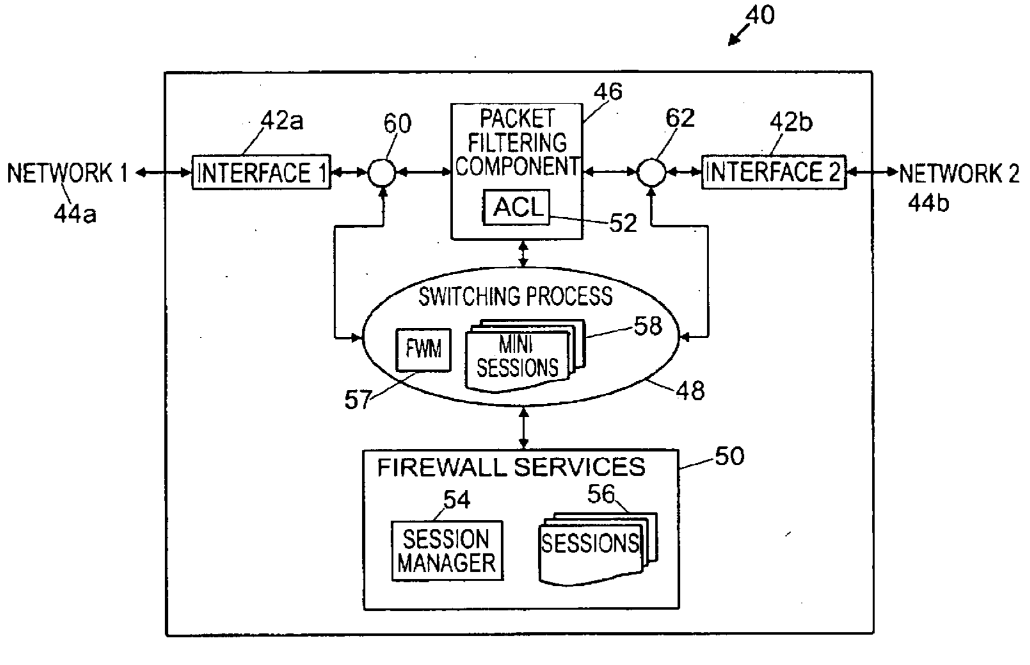
* cited by examiner

Primary Examiner—Gregory Morse
Assistant Examiner—Matthew Heneghan
(74) *Attorney, Agent, or Firm*—Sierra Patent Group, Ltd.

(57) **ABSTRACT**

A firewall system and method which optimizes the performance of the firewall process by reducing overhead associated with ACL verification and firewall application-level authorization. The firewall system comprises a session manager operating in the firewall services component and a firewall module operating in the switching process component. In one embodiment, the firewall module is configured to provide certain “non-application” level inspection of data packets and update the context of “sessions” associated with the data packets without sending the packets to the firewall services component using session information provided by the session manager.

25 Claims, 6 Drawing Sheets



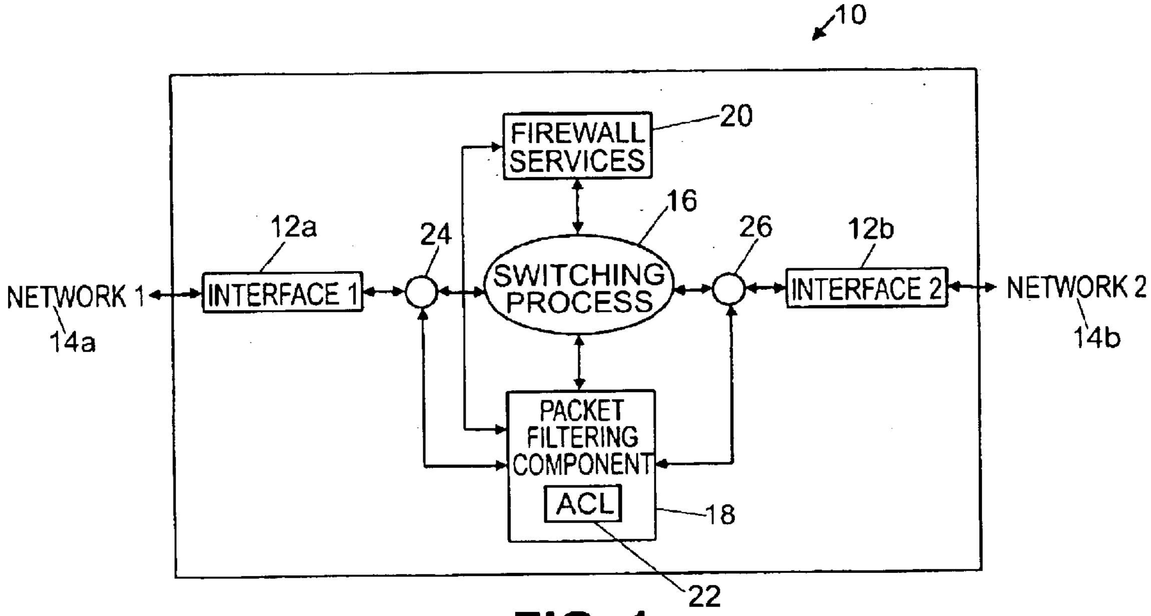


FIG. 1
PRIOR ART

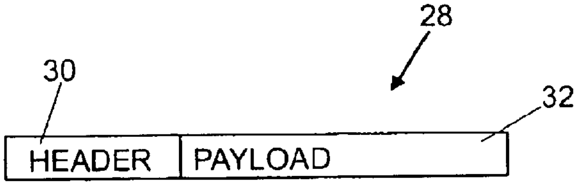


FIG. 2
PRIOR ART

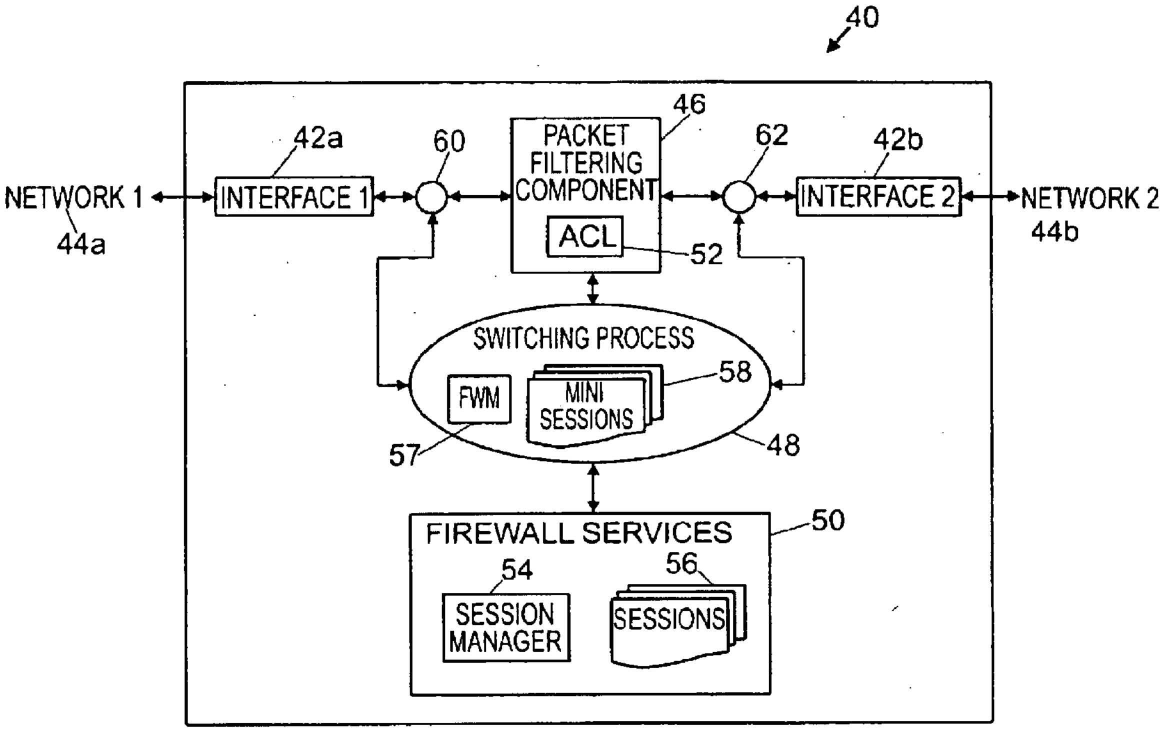

40
NETWORK 1
44a
INTERFACE 1
42a
60
PACKET FILTERING COMPONENT
46
ACL
52
62
INTERFACE 2
42b
NETWORK 2
44b
SWITCHING PROCESS
FWM
57
MINI SESSIONS
58
48
FIREWALL SERVICES
50
SESSION MANAGER
54
SESSIONS
56

FIG. 3

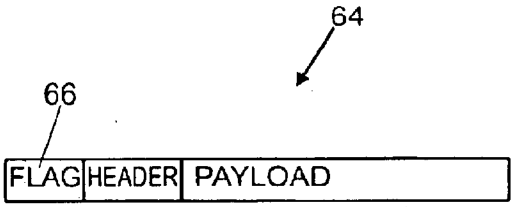

64
66
FLAG
HEADER
PAYLOAD

FIG. 4

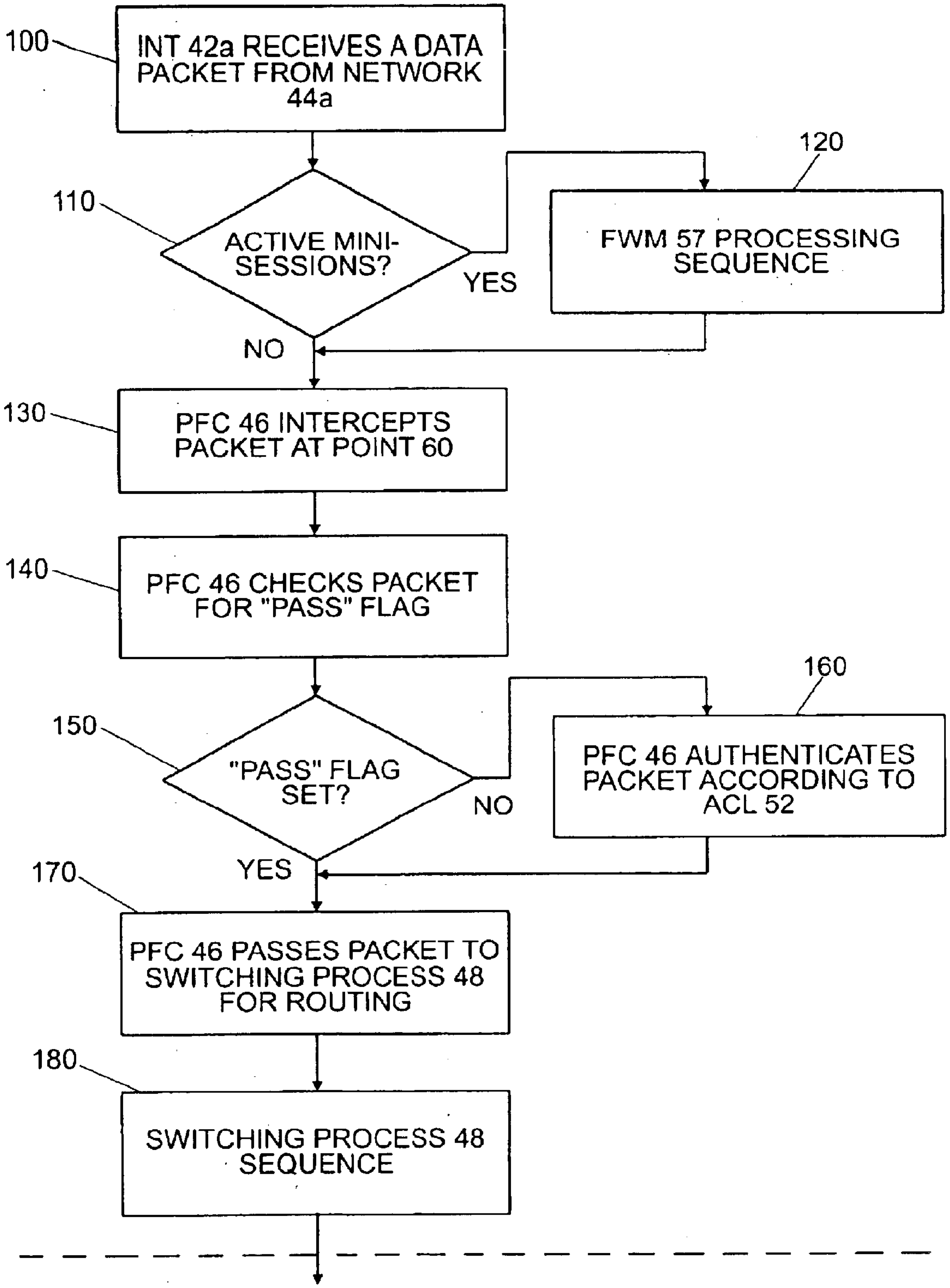

100
INT 42a RECEIVES A DATA PACKET FROM NETWORK 44a
110
ACTIVE MINI-SESSIONS?
YES
NO
120
FWM 57 PROCESSING SEQUENCE
130
PFC 46 INTERCEPTS PACKET AT POINT 60
140
PFC 46 CHECKS PACKET FOR "PASS" FLAG
150
"PASS" FLAG SET?
NO
YES
160
PFC 46 AUTHENTICATES PACKET ACCORDING TO ACL 52
170
PFC 46 PASSES PACKET TO SWITCHING PROCESS 48 FOR ROUTING
180
SWITCHING PROCESS 48 SEQUENCE

FIG. 5A

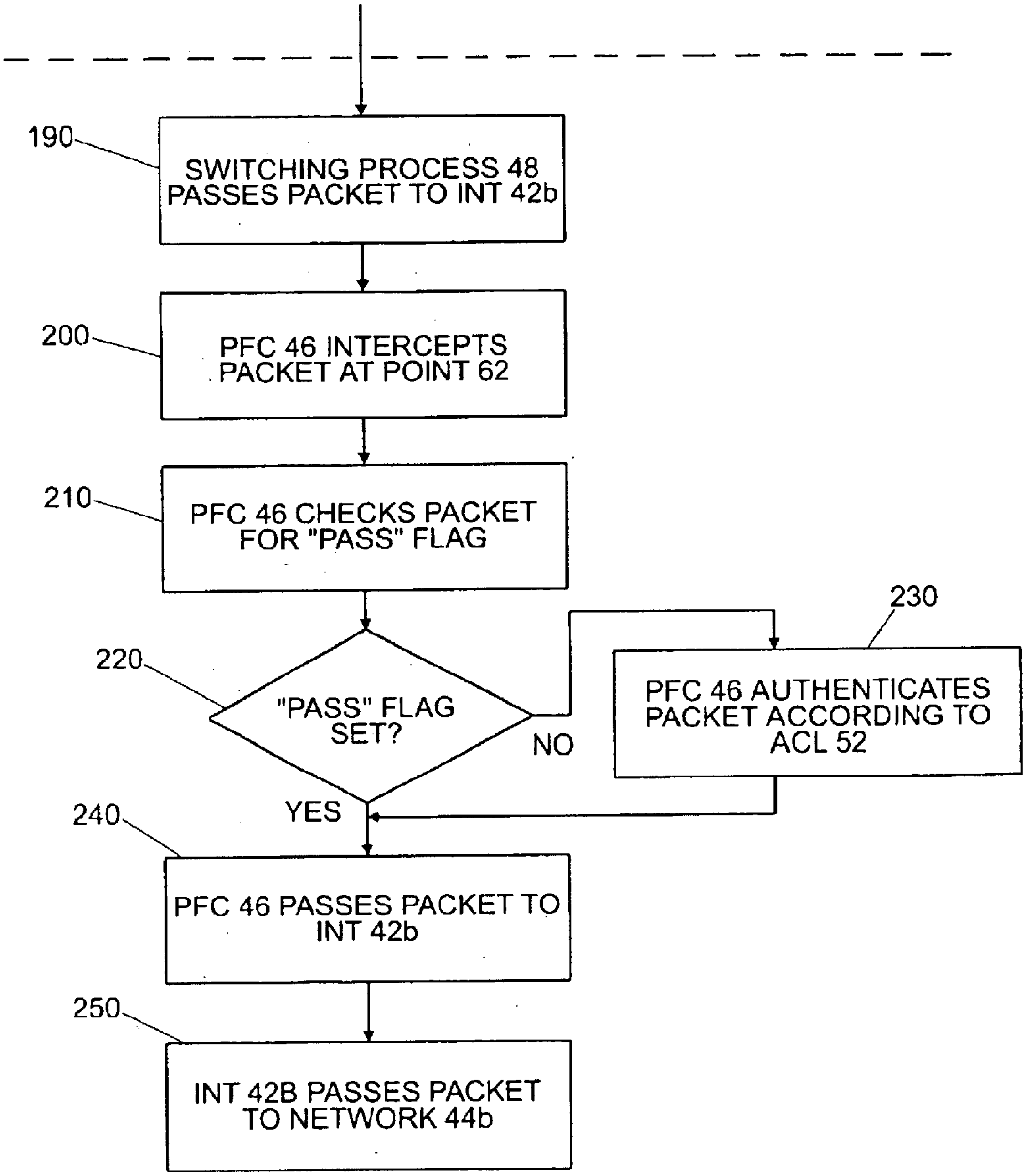

190
SWITCHING PROCESS 48 PASSES PACKET TO INT 42b
200
PFC 46 INTERCEPTS PACKET AT POINT 62
210
PFC 46 CHECKS PACKET FOR "PASS" FLAG
220
"PASS" FLAG SET?
NO
230
PFC 46 AUTHENTICATES PACKET ACCORDING TO ACL 52
YES
240
PFC 46 PASSES PACKET TO INT 42b
250
INT 42B PASSES PACKET TO NETWORK 44b

FIG. 5B

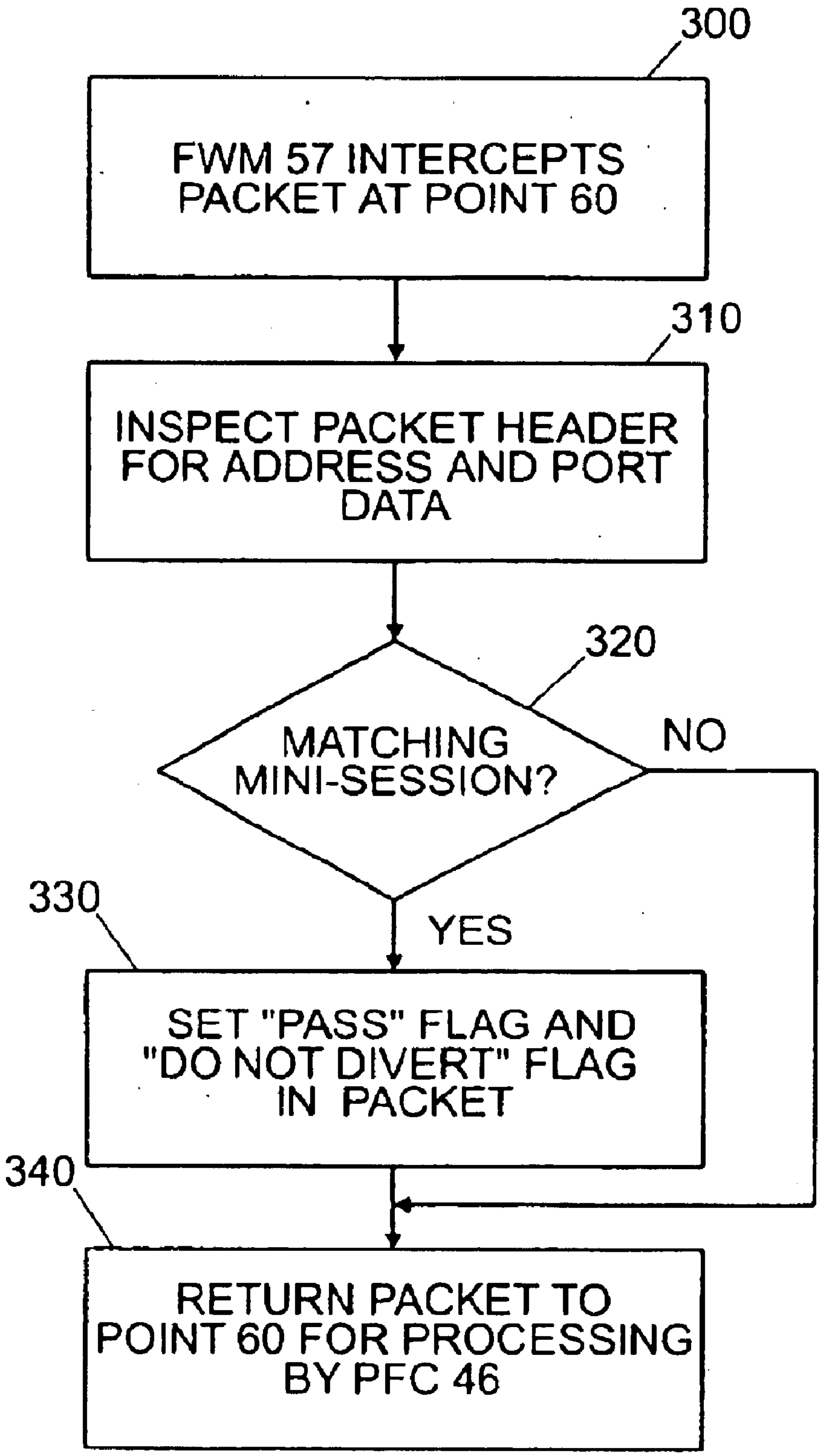

300
FWM 57 INTERCEPTS PACKET AT POINT 60
310
INSPECT PACKET HEADER FOR ADDRESS AND PORT DATA
320
MATCHING MINI-SESSION?
NO
YES
330
SET "PASS" FLAG AND "DO NOT DIVERT" FLAG IN PACKET
340
RETURN PACKET TO POINT 60 FOR PROCESSING BY PFC 46

FIG. 6

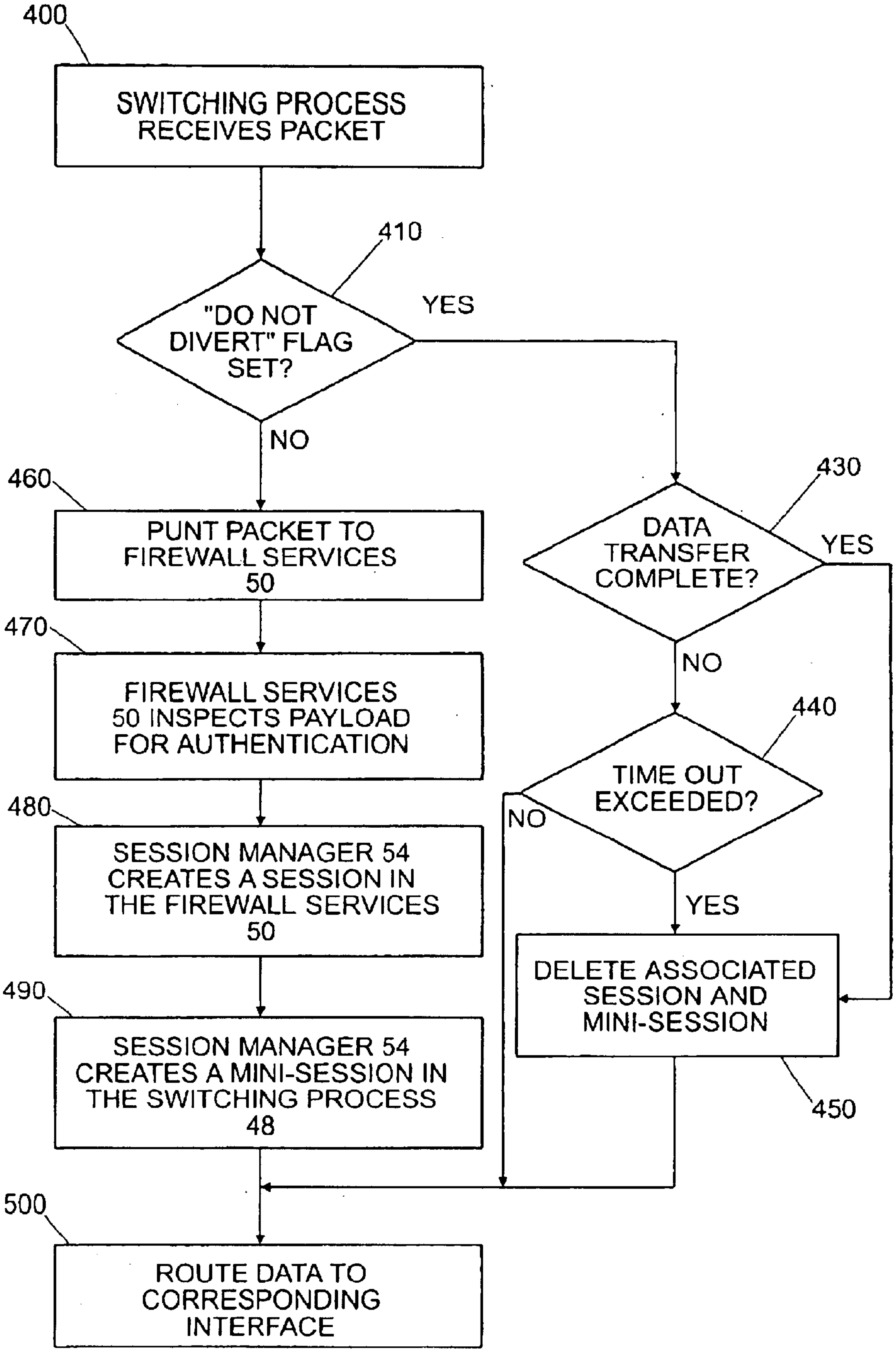

400
SWITCHING PROCESS RECEIVES PACKET
410
"DO NOT DIVERT" FLAG SET?
YES
NO
460
PUNT PACKET TO FIREWALL SERVICES 50
470
FIREWALL SERVICES 50 INSPECTS PAYLOAD FOR AUTHENTICATION
480
SESSION MANAGER 54 CREATES A SESSION IN THE FIREWALL SERVICES 50
490
SESSION MANAGER 54 CREATES A MINI-SESSION IN THE SWITCHING PROCESS 48
500
ROUTE DATA TO CORRESPONDING INTERFACE
430
DATA TRANSFER COMPLETE?
YES
NO
440
TIME OUT EXCEEDED?
NO
YES
DELETE ASSOCIATED SESSION AND MINI-SESSION
450

FIG. 7

METHOD AND APPARATUS FOR OPTIMIZING FIREWALL PROCESSING

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention pertains generally to firewall systems. More particularly, the invention is a firewall system and method which optimizes the performance of the firewall process by reducing overhead associated with ACL verification and firewall application-level authorization.

2. The Prior Art

Firewalls are known in the art. In general, a firewall is a combination of hardware and software which limits the exposure of a computer or group of computers to an attack from outside. The most common use of a firewall is on a local area network (LAN) connected to the global information network, known as the Internet. Without a firewall, anyone on the Internet could theoretically connect to the corporate LAN and retrieve and/or transmit information to computers on the LAN. A firewall provides services which enforce a boundary between two or more networks. In the above example, a firewall would enforce a boundary between the LAN and the Internet.

A traditional firewall is implemented through a combination of hosts and routers. A router can control traffic at the packet level, allowing or denying packets based on the source/destination address or the source/destination port number. A host (or application gateway), on the other hand, can control traffic at the application level, allowing control based on a more detailed and protocol-dependent examination of the traffic. Often, a router can be configured to provide firewall capability.

FIG. **1** depicts a block diagram of a firewall (or router with firewall capbilities) device **10** according to the prior art. Firewall **10** is shown having interface **1** (designated as **12***a*) and interface **2** (designated as **12***b*), interface **1** (**12***a*) connected to a network 1 (**14***a*) and interface **2** (**12***b*) connected to network **2** (**14***b*). As a data packet is communicated from network **1** (**14***a*) to network **2** (**14***b*) or from network **2** (**14***b*) to network **1** (**14***a*), the data is intercepted and is authorized or denied data communication based on a plurality of configuration settings as is well known in the art.

The firewall device **10** includes a plurality of services to carry out the operation of authorizing data traffic through the device **10**. More particularly, the firewall device **10** includes a switching process component (or router) **16**, a packet filtering component **18**, and a firewall services component **20**. Switching process **16** handles traffic connections associated with interfaces **12***a*, **12***b*, and routes data according to designated addresses.

The packet filtering component **18** filters data packets based on a set of rules defined in an associated Access Control List, designated ACL **22**. The ACL **22** contains static as well as dynamic settings. Static settings are normally provided by a user in a configuration file. For example, the ACL **22** may define a set of IP (Internet Protocol) addresses that are permitted to communicate through firewall device **10**. Dynamic settings are normally provided by the firewall services to enable certain communications, including return acknowledgement signals, for example.

The firewall services **20** provide authentication on an application level, providing among other things, protocol dependent inspection and authentication. As noted above, the firewall service **20** also configures the ACL **22** to allow certain communications to pass through the device **10**.

The following example illustrates the operation of a prior art firewall device. FIG. **2** shows the structure of a typical data packet **28** transmitted through the firewall device. As is known, the data packet **28** will include a header portion **30**, and a data payload component **32**. The header portion **30** includes among other things, address information (such as the destination address, for example).

The data payload component **32** includes additional information such as User ID and protocol information, among other things.

When a data packet is communicated from network **14***a* to network **14***b*, for example, the data packet enters interface **12***a* from network **14***a*. The packet filtering component **18** intercepts the data packet at point **24** and determines whether the data packet is authorized to enter the router **10** via the interface **12***a* based on the set of rules defined in the ACL **22**. Typically, the header is inspected to determine source and/or destination address information. If so authorized, the data packet is communicated back to point **24** and to the switching process **16** for routing to the appropriate interface.

The switching process **16** receives the data packet and "diverts" the data packet to the firewall services **20** for inspection and authorization. As noted above, the firewall services component **20** authenticates the data packet based on a set of protocol-dependent rules. In this way, the payload component **32** of the data packet is typically inspected to see if communication is authorized. If so authorized by the firewall services **20**, the data packet is sent back to the switching process **16**, which then communicates the data packet to the interface **12***b* via point **26**. As noted above, the firewall services component **20** may also configure one or more settings within the ACL **22** to allow certain communications (return acknowledgments, for example) to pass through the router **10**.

At point **26**, the data packet communicated by the switching process **16** is again intercepted by the packet filtering component **18** to determine whether the data packet is authorized to exit interface **12***b* based of the set of rules defined in the ACL **22**. If so authorized, the switching process **16** transmits the data packet to the interface **12***b*, via point **26**, which then communicates the data packet to network **14***b* where the packet is further processed.

As described above, the prior art method of firewall processing involves a plurality of security authorization steps. For each data packet that is communicated through the firewall device, the authorization steps described above are carried out. In the above example firewall **10** having 2 ports **14***a*, **14***b*, two ACL authentication processes are carried out by the packet filtering component **18**, one for each port. Additionally, the switching process **16** diverts the packet to the firewall services **20** for application-level authentication.

While providing security, there are performance penalties associated with the above described authorization processes for a firewall. For example, because the switching process **16** operates in a different address space from the firewall services component **20**, the router device **10** suffers the overhead associated with "context switching" when a data packet is "diverted" from the switching process **16** to the firewall services **20** for inspection and authorization.

Prior art firewalls "divert" each data packet handled by switching process **16**. However, since data communication transactions often involve the transfer of a plurality of data packets (rather than a single packet), the need for authorizing (and therefore diverting) each and every packet may be unnecessary, once the first in a series of associated data packets has been authorized. This is particularly evident in a data transfer, as opposed to a control transfer.

For example, an FTP transfer of a file may involve the transfer or a plurality of packets. If the first packet is authorized between a source and a destination, then the remaining associated packets would also be authorized. However, under present firewall solutions, each of the remaining packets would be diverted and authorized, and thus the overhead associated with context-switching is realized for each of the associated packets for the duration of the file transfer.

Also as noted above, the packet filtering component **18** carries out authorization based on the rules provided in the corresponding ACL **22**. This authorization is carried out for each packet processed by the packet filtering component **18**. As noted above, the packet is checked upon entering a port and upon exiting a port, thus incurring additional performance penalties.

Accordingly, there is a need for a method and apparatus which provides firewall security processing which minimizes the overhead with context switching and optimizes overall firewall performance. The present invention satisfies these needs, as well as others, and generally overcomes the deficiencies found in the background art.

BRIEF DESCRIPTION OF THE INVENTION

The present invention is a system and method for optimizing firewall performance, which reduces the overhead associated with firewall protocol inspection and packet filtering authorization. The invention further relates to machine readable media on which are stored embodiments of the present invention. It is contemplated that any media suitable for retrieving instructions is within the scope of the present invention. By way of example, such media may take the form of magnetic, optical, or semiconductor media. The invention also relates to data structures that contain embodiments of the present invention, and to the transmission of data structures containing embodiments of the present invention.

The present system operates in a conventional firewall (or router having firewall capabilities) device having conventional hardware components, such as a central processor unit (CPU), memory, and input/output devices. The firewall device will typically further include a plurality of communication interfaces (interfaces), such as Ethernet ports and/or serial ports, for example. As such, the firewall device inspects and authenticates communication carried out between the various ports of the firewall device.

According to one embodiment of the present invention, the firewall system operates in a firewall device having a plurality of communication interfaces, a packet filtering component coupled to each of the interfaces, a switching component coupled to each of the interfaces, and a firewall services component coupled to the switching process. The firewall system comprises a session manager operating in the firewall services component. The session manager is structured and configured to instantiate a plurality of sessions in the firewall services component and a plurality of mini-sessions in the switching process component. Each of the sessions has context information captured from the header and payload information of packets of the session. Each of the mini-sessions corresponds to a session and includes header information related the corresponding data transfer within the firewall device. The mini-sessions are managed by a firewall module (or mini-session manager) residing and operating in the switching process/component. This firewall module is initiated by the session manager component and is configured to maintain data in the mini-sessions. The firewall module further determines whether to send packets from the switching process to the firewall process based on the data in the mini-sessions.

According to another embodiment of the present invention, the method for optimizing firewall processing comprises providing a session manager in the firewall services component, providing a firewall module in the switching process/component, instantiating a session, by the session manager, for data transfers within the firewall device, the sessions having header and payload information related to data transfers within the firewall device, and instantiating a mini-session, by the session manager, corresponding to the instantiated session, the mini-session having header information related to data transfers within the firewall device.

An object of the invention is to provide a firewall system and method for optimizing firewall processing that overcomes the deficiencies in the prior art.

Another object of the invention is to provide a firewall system and method for optimizing firewall processing that reduces the overhead associated with context-switching in firewall authentication.

Yet another object of the invention is to provide a firewall system and method for optimizing firewall processing that reduces the overhead associated with access control list authentication.

Further objects and advantages of the invention will be brought out in the following portions of the specification, wherein the detailed description is for the purpose of fully disclosing the preferred embodiment of the invention without placing limitations thereon.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be more fully understood by reference to the following drawings, which are for illustrative purposes only.

FIG. **1** is a block diagram depicting a firewall system according to the prior art.

FIG. **2** is a block diagram depicting the structure of a conventional data packet.

FIG. **3** is a block diagram depicting a firewall system in accordance with the present invention.

FIG. **4** is a block diagram depicting the structure of a data packet according to present invention.

FIGS. **5***a*, and **5***b* is a flow chart generally showing the acts associated with carrying out firewall system processes in accordance with the present invention.

FIG. **6** is a flow chart generally showing the acts associated with the mini-session processes in accordance with the present invention.

FIG. **7** is a flow chart generally showing the acts associated with the firewall services processes in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Persons of ordinary skill in the art will realize that the following description of the present invention is illustrative only and not in any way limiting. Other embodiments of the invention will readily suggest themselves to such skilled persons having the benefit of this disclosure.

Referring more specifically to the drawings, for illustrative purposes the present invention is embodied in the apparatus shown FIG. **3** and FIG. **4** and the method outlined in FIG. **5***a* through FIG. **7**. It will be appreciated that the apparatus may vary as to configuration and as to details of the parts, and that the method may vary as to details and the order of the acts, without departing from the basic concepts as disclosed herein. The invention is disclosed generally in terms of a firewall system and method, although numerous other uses for the invention will suggest themselves to persons of ordinary skill in the art.

Referring now to FIG. **3**, there is generally shown a block diagram depicting a firewall system in accordance with the present invention. The present system operates in a conventional firewall or router device **40** having conventional hardware components (not shown), such as a central processor unit (CPU), memory, and input/output devices. The firewall device will typically further include a plurality of communication interfaces (interfaces) **42***a*, **42***b*, which may be Ethernet ports and/or serial ports, for example. Interface **42***a* is operatively coupled for communication with a first network designated **44***a*, and interface **42***b* is operatively coupled for communication with a second network designated **44***b*. Network **44***a*, **44***b* comprise convention networks, such a local area network (LAN), wide area network (WAN), and/or the Internet, for example. Under this arrangement, the firewall device carries out the operation of inspecting and authenticating communication carried out between the various ports **42***a*, **42***b* of the firewall device.

The firewall system of the present invention is generally embodied in software executing and operating in the firewall **40** and carries out the operation described herein. The firewall system generally includes a packet filtering component **46**, a switching process component **48**, and a firewall services component **50**.

The packet filtering component **46** is a software module coupled for communication to the interfaces **42***a*, **42***b* of the firewall device **40**. The packet filtering component **46** inspects packets entering and exiting the communication interfaces **42***a*, **42***b* to determine whether transfer is authorized according to certain port and address settings maintained in an "Access Control List" (ACL **52**). The packet filtering component **46** of the present invention, however, does not perform ACL verification for each packet processed through a port as is carried out in prior art firewall system. Rather, the packet filtering component **46** first checks the data packet for a "pass" flag which may be set by the firewall system. If the "pass" flag is set, the packet filtering component **46** bypasses ACL verification. If the "pass" flag is not set, ACL verification is carried out conventionally. This process is described more fully below in conjunction with FIG. **5***a* and FIG. **5***b*.

The switching process component **48** is a software module coupled for communication to the interfaces **42***a*, **42***b* of the firewall device. The switching process component routes packets between the interfaces **42***a*, **42***b* of the firewall device, according to address and port information, as is known in the art.

The switching process further comprises a "firewall module" (or mini-session manager), designated FWM **57**. The FWM **57** is a software module operating in the same memory space as switching process **48** which performs session tracking operations, including "non-application-level" inspection of data packets, and "diverting" data packets to the firewall services **50** for inspection and authorization thereof when appropriate as described in further detail in conjunction with FIG. **7** below. The FWM **57** is initiated during the startup of the device **40** by the firewall services **50**.

The FWM **57** is coupled to one or more "mini-sessions" (generally designated as **58**) which are described in more detail below in conjunction with FIG. **5***a* through FIG. **7**. The "mini-sessions" are instantiated software modules residing in the same address space as the switching process component **48** which contain session information data used by the FWM **57** for processing data packets.

The firewall services component **50** is a software module coupled for communication to the switching process component **48** which inspects and authorizes packets according to information in the payload component of the packet, the process of which is known in the art.

According to the present invention, once a data packet from a first address is authorized for transfer to a second address, the firewall system authorizes "future" data transfers between the first and second addresses to complete the data transfer so authorized. In this way, the system provides a means for authorizing such "future" data transfers without ACL authorization and firewall "application-level" authorization in order to complete the data transfer transaction. To this end, the firewall system of the present invention provides a session manager **54** operating within the firewall services component **50**. The session manager **54** is configured to create, manage and delete a plurality of sessions **56** in the firewall services and a plurality of "mini-sessions" **58** in the switching process component **48**.

In general, when the firewall services **50** encounters a data packet (an FTP data transfer, for example), as opposed to a control packet (a login command, for UL example), for transfer within the firewall **40**, the session manager **54** instantiates a corresponding session **56**, for storing among other things, header data (addresses and port information, for example) as well as payload data (user id and communication protocol, for example) to track the data transfer. This instantiated session **56** is maintained in the firewall services **50** by the session manager **54** until the data transfer is either completed or exceeds a predetermined timeout (a period of time during which no corresponding data transfer is carried out). The session manager **54** further instantiates a corresponding "mini-session" **58** in the switching process **48** which contains, among other things, corresponding header data used by the FWM **57**. In this manner, for each session **56** maintained by the session manager **54**, there is a corresponding mini-session **58** in the switching process **48**.

The processes of the FWM **57** in conjunction with the mini-sessions **58** are described further below in conjunction with FIG. **6**. In general, the FWM **57** is configured to intercept data packets entering the interfaces **42***a*, **42***b* of the firewall **40**, preferably before the packet filtering component **46** intercepts the packet for ACL verification. The FWM **57** then compares the data in the header component of the intercepted packet to corresponding data in each mini-session **58** to determine whether the packet is authorized to pass without authorization. A packet is authorized when a source and destination address and port number specified in the header matches corresponding source and destination address port number in one of the mini-sessions. If so, the FWM **57** sets a "pass" flag in the packet to communicate that ACL verification is to be bypassed and a "do not divert" flag to communicate that firewall "application" level authorization is to be bypassed.

The FWM **57** is also configured to update the connection state "in-place" (i.e., without sending the packets to the firewall process). This ability is particularly important to reduce the overhead involving context switching between different processes, i.e., the switching process and the firewall services component process. Without this "in-place" inspection, all packets would have to be sent to an external process (e.g., the firewall services **50**), causing additional overhead and latency. The scope of this invention does not limit the capability of the mini-session manager just to a subset of the connections or sessions. While the illustrative FWM **57** in FIG. **3** performs non-application-level inspection, it is possible to create a firewall module that has the ability to perform application-level inspection and therefore eliminate the need to transfer packets to an external process once the inspection of a new connection begins. However, to provide speed at the switching process **48** level, the FWM **57** described in FIG. **3** is configured to carry out non-application-level inspection.

FIG. **4** depicts an example data structure for a data packet **64** where flag bits may be set in slot **66** by the FWM **57** to define the "pass" state and the "do not divert" state as described above. For example, slot **66** may comprise two bits, the first bit ("pass" bit) for defining "pass" state and the second bit ("do not divert" bit) for defining the "do not divert" state. While depicted as the first slot of the data structure **64**, the communication of a "pass" flag and "do not pass" flag may be carried out using various other slot arrangements as is known in the art.

Once a data transfer is completed, the associated session **56** and mini-session **58** is deleted. In general, the FWM **57** monitors communications between the interfaces **42***a*, **42***b* for signals indicating a completed transfer as is known in the art. Once this signal is so detected, a "delete" request is sent to the session manager **54** by the FWM **57**. In response, to the "delete" request, the session manager **54** deletes corresponding session **56** and mini-session **58**. Sessions **56** and corresponding mini-sessions **58** are also deleted if the data transfer associated therewith is idle (not communicating) for a predetermined period of time ("timeout" period).

The method and operation of invention will be more fully understood with reference to the flow charts of FIG. **5***a* through FIG. **7**, as well as FIG. **3** and FIG. **4**. The order of actions as shown in FIG. **5***a* through FIG. **7** and described below is only exemplary, and should not be considered limiting.

Referring next to FIG. **5***a* and FIG. **5***b*, as well as FIG. **2** through FIG. **4**, there is generally shown the acts associated with the firewall system processes of the present invention. FIG. **5***a* and FIG. **5***b* depict the example situation where a data packet is communicated from Network **1** to Network **2**. An analogous series of acts are carried out for data packets communicated from Network **2** to Network **1**.

At box **100**, Interface **42***a* receives a data packet from Network **1** for communication to Network **2**. The data packet will typically be structured as packet **28** (FIG. **2**) having a header component and a payload component. In general, the header component includes source and destination address information as well as port information. Box **110** is then carried out.

At box **110**, the FWM **57** operating in the switching process **48** ascertains whether there are any active mini-sessions **58** operating therein. As described above, mini-deleted sessions **58** are instantiated and deleted during operation of the firewall device **40**. If there are active mini-sessions **58**, box **120** is carried out. Otherwise, box **130** is carried out.

At box **120**, the FWM **57** processing sequence is carried out. This sequence is described in more detail in conjunction with FIG. **6** below. After the FWM **57** processing sequence is completed, box **130** is then carried out.

At box **130**, the packet filtering component (PFC) **46** intercepts the packet at point **60** before the switching process **48** carries out routing tasks (described in box **180** below). Box **140** is then carried out.

At box **140**, the PFC **46** inspects the intercepted data packet to determine whether a "pass" flag is set. In this way, the PFC **46** checks a predetermined slot in the data structure. For example, in FIG. **4**, flag slot **66** may contain a "1" bit in the appropriate bit location (i.e., the "pass" bit) to indicate a "pass". Diamond **150** is then carried out.

At diamond **150**, the PFC **46** determines whether the "pass" flag is set in the intercepted data packet. If the "pass" flag is set, box **170** is then carried out to bypass the ACL verification. Otherwise, box **160** is carried out.

At box **160**, the PFC **46** authenticates the packet entering interface **42***a* according to the settings provided in the ACL **52** as is known in the art. This typically involves comparing the header of the packet to corresponding settings in the ACL **52**. Packets which are authorized are further processed, while unauthorized packets are dropped. Box **170** is then carried out.

At box **170**, the PFC **46** returns the packet for processing by the switching component **48**. Box **180** is then carried out.

At box **180**, the switching process **48** sequence is carried out. In general, this sequence includes, among other things, routing the packet to the appropriate interface as well as verifying the packet with the firewall services component **50** when appropriate. This sequence is described more fully below in conjunction with FIG. **7**. Box **190** is then carried out.

At box **190**, the switching process **48** routes the packet to the appropriate interface, which in the present example is interface **42** (for communication to Network **44***b*). Box **200** is then carried out.

At box **200**, the packet filtering component (PFC) **46** intercepts the packet at point **62** before the packet is received by interface **42***b*. Box **210** is then carried out.

At box **210**, the PFC **46** inspects the intercepted data packet to determine whether a "pass" flag is set. As described above in box **140**, the PFC **46** check a predetermined slot in the data structure. Diamond **220** is then carried out.

At diamond **220**, the PFC **46** determines whether the "pass" flag is set in the intercepted data packet. If the "pass" flag is set, box **240** is then carried out to bypass ACL verification. Otherwise, box **230** is carried out.

At box **230**, the PFC **46** authenticates the packet exiting via interface **42***b* according to the settings provided in the ACL **52**. As noted above, ACL verification typically involves comparing the header of the packet to corresponding settings in the ACL **52**. Packets which are authorized are further processed, while unauthorized packets are dropped. Box **240** is then carried out.

At box **240**, the PFC **46** returns the packet to interface **42***b* for further processing. Box **250** is then carried out.

At box **250**, the interface **42***b* passes the data packet to network **44***b* where the data packet is further processed.

Referring now to FIG. **6**, as well as FIG. **2** through FIG. **5***b*, there is generally shown the acts associated with the Firewall Manager (FWM **57**) processes in accordance with the present invention. This sequence is carried out during box **120** of FIG. **5***b*.

At box **300**, the FWM **57** intercepts an "entering" packet. In the present illustrative case the packet entering firewall device **10** via interface **42***a* is intercepted at point **60**.

Preferably, FWM **57** intercepts the packet prior to interception by PFC **46** (box **130** of FIG. **5***a*). Box **310** is then carried out.

At box **310**, the FWM **57** inspects the header component of the data packet to ascertain the address (source and destination) and port information contained therein. Box **320** is then carried out.

At box **320**, the address and port information obtained in box **310** is compared with the corresponding information in the plurality of mini-sessions **58**. If a match is established between the packet and a corresponding mini-session **58** according to address and port information, then box **330** is carried out. Otherwise box **340** is carried out.

At box **330**, the intercepted packet has a matching mini-session **58**. A match indicates that the intercepted packet is authorized for transfer without ACL verification and firewall application-level inspection through the firewall device **40**. Thus, a "pass" flag is set with the packet to communicate this authorization. For example, as shown in FIG. **4**, the "pass" bit of flag slot **66** may be set with a "1" bit to indicate a "pass" flag. Conversely, a "0" bit may be used in the "pass" bit of slot **66** to indicate a lack of a "pass" flag. As described above, the PFC **46** will bypass ACL verification if this "pass" flag is set.

The FWM **57** also determines if the intercepted packet has a matching mini-session which can be updated or inspected "in-place". In general, sessions (and mini-sessions) can be updated in-place if application-level inspection is not required. If the mini-session can be updated in-place, a "do not divert" flag (e.g., a "1" bit in the "do not divert" bit of slot **66**) is set to communicate that the switching process **48** will bypass firewall application level inspection. Box **340** is then carried out.

At box **340**, the packet is returned to point **60** (the interception point) for further processing by the PFC **46**.

Referring now to FIG. **7**, as well as FIG. **2** through FIG. **6**, there is generally shown the acts associated with the firewall services processes in accordance with the present invention. This sequence is carried out during box **180** of FIG. **5***a*.

At box **400**, the switching process receives a data packet for inspection and routing, typically from the PFC **46**. Diamond **410** is then carried out.

At diamond **410**, the switching process inspects the data packet to determine whether the "do not divert" flag is set. As described above, the "do not divert" flag is set by the FWM **57** if the intercepted packet has a matching mini-session which can be updated or inspected "in-place" (box **330** of FIG. **6**). In general, sessions can be updated in-place if application-level inspection is not required. As noted above, the FWM **57** may be configured to provide application-level inspection, but the switching process **48** suffers a performance penalty for carrying out such inspection. If the "do not divert" flag is set, diamond **430** is carried out, otherwise box **460** is carried out.

At diamond **430**, the mini-session (as associated session) can be updated in-place. As such, the intercepted packet is authorized for transfer without ACL verification and firewall application-level inspection through the firewall device **40**. Thus, firewall application-level inspection (box **460** and **470**) is bypassed. Instead the packet is inspected to ascertain whether the transfer is completed. This check is typically carried out by detecting a signal indicating the end of the data. If the data transfer is completed, box **450** is carried out. Otherwise diamond **440** is carried out.

At diamond **440**, the FWM **57** tracks whether a predetermined timeout period for data transfer has been exceeded using conventional methods known in the art, and as described above. The timer to track this timeout period may be maintained by the FWM **57** or by the session manager **54**. If the timeout period is exceeded, box **450** is carried out. Otherwise box **500** is carried out.

At box **450**, the data transfer has either completed or exceeded the timeout threshold. As such the session and mini-session corresponding to the data transfer are deleted, typically by communicating a request to the session manger **54** which then carries out the deletion of the corresponding session and mini-session. Box **500** is then carried out.

At box **460**, the "do not divert" flag is not set for the intercepted packet. Accordingly, the switching process **48** diverts the packet to the firewall services **50** for inspection and authorization of the packet. Box **470** is then carried out.

At box **470**, the firewall services component **50** inspects the payload component of the data packet for authentication using conventional application-level inspection means. If the packet is authorized, box **480** is carried out. Otherwise the packet is deleted.

At box **480**, the session manager **44** instantiates a session **56** in the firewall services **50** corresponding to the current data transfer. As note above, the created session includes header data (addresses and port information, for example) as well as payload data (user id and communication protocol, for example) to track the data transfer associated with the currently inspected packet. Box **490** is then carried out.

At box **490**, the session manger **44** instantiates a corresponding mini-session **58** in the switching process to authorize future data transfers associated with the presently inspected packet. It is noted that as packets are processed herein, the session and associated mini-session are created only once if they do not exists, and not for every packet associated with the corresponding session. Box **500** is then carried out.

At box **500**, the packet is then routed to the appropriate interface according to the address information contained in the header of the packet. In the present example, the data packet is communicated from network **44***a* to network **44***b*, and thus is routed to interface **42***b* via point **62** for further processing.

Accordingly, it will be seen that this invention provides a firewall system and method which optimizes the performance of the firewall process by reducing overhead associated with ACL verification and firewall-application level authorization. Although the description above contains many specificities, these should not be construed as limiting the scope of the invention but as merely providing an illustration of the presently preferred embodiment of the invention. Thus the scope of this invention should be determined by the appended claims and their legal equivalents.

What is claimed is:

1. In a firewall device having a plurality of communication interfaces, a packet filtering component coupled to each of the interfaces, a switching process component coupled to each of the interfaces, and a firewall services component coupled to the switching process component, a firewall system comprising:

a) a session manager operating in said firewall services component, said session manager structured and configured to instantiate a plurality of sessions in said firewall services component and a plurality of mini-sessions in said switching process component, each of said plurality of sessions having header and payload information related to a corresponding data transfer within the firewall device, each of said plurality of mini-sessions corresponding to a session and including header information related the corresponding data transfer within the firewall device, wherein said plurality of mini-sessions comprises instantiated software modules residing in the same address space as said switching process component; and

b) a firewall module operating in said switching process coupled to said plurality of mini-sessions, said firewall module configured to intercept data packets received into the interfaces, said firewall module further configured to track session context of said data packets.

2. The firewall system of claim **1**, wherein said session manager is further structured and configured to manage said plurality of sessions and said plurality of mini-sessions.

3. The firewall system of claim **1**, wherein said session manager is further structured and configured to delete said plurality of sessions and said plurality of mini-sessions.

4. The firewall system of claim **1**, wherein said firewall module is further configured to intercept data packets before reception by said packet filtering component, said firewall module further configured to set a "pass" flag in data packets according matching header information in intercepted data packets and said header information in said plurality of mini-sessions.

5. The firewall system of claim **4**, wherein said packet filtering component is configured to bypass "Access Control List" authorization of data packets having a "pass" flag.

6. The firewall system of claim **1**, wherein said firewall module is further configured to intercept data packets before reception by said packet filtering component, said firewall module further configured to set a "do not divert" flag in data packets when packet inspection of said intercepted data packets does not require application-level inspection.

7. The firewall system of claim **6**, wherein said firewall module is configured to bypass authorization of data packets having a "do not divert" flag with said firewall services component.

8. In a firewall device having a plurality of communication interfaces, a packet filtering component coupled to each of the interfaces, a switching process component coupled to each of the interfaces, and a firewall services component coupled to the switching process component, a method for optimizing firewall processing comprising:

a) providing a session manager in the firewall services component;

b) providing a firewall module in the switching process component;

c) instantiating a session, by said session manager, for data transfers within the firewall device, said sessions having header and payload information related to data transfers within the firewall device; and

d) instantiating a mini-session, by said session manager, corresponding to said instantiated session, said mini-session having header information related to data transfers within the firewall device, wherein said mini-session comprises instantiated software modules residing in the same address space as said switching process component.

9. The method of claim **8**, further comprising:

a) intercepting data packets having a header and a payload component, by said firewall module, before reception by the packet filtering component; and

b) setting a "pass" flag in the intercepted data packets when said header component is the intercepted data packets matches said header information in said mini-session.

10. The method of claim **8**, further comprising:

a) checking data packets for a "pass" flag, by said packet filtering component; and

b) bypassing "access control list" check, if a "pass" flag is found in said checked data packets.

11. The method of claim **8**, further comprising:

a) intercepting data packets having a header and a payload component, by said firewall module, before reception by the packet filtering component; and

b) setting a "do not divert" flag in the intercepted data packets when packet inspection does not require application-level inspection.

12. The method of claim **8**, further comprising:

a) checking data packets for a "do not divert" flag, by said firewall module; and

b) bypassing "access control list" check, if a "do not divert" flag is found in said checked data packets.

13. The method of claim **8**, further comprising bypassing authorization with the firewall services component, by the firewall module, for data packets header information matching header information in said mini-sessions.

14. The method of claim **8**, further comprising deleting said session and associated mini-session when data transfer associated with said sessions and mini-session is completed.

15. The method of claim **8**, further comprising deleting said session and associated mini-session when data transfer associated with said sessions and mini-session is idle past a predetermined timeout period.

16. The method of claim **8**, further comprising updating context of said mini-session, by said firewall module, without sending packets to said firewall services component.

17. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform a method for optimizing firewall processing in a firewall device having a plurality of communication interfaces, a packet filtering component coupled to each of the interfaces, a switching process component coupled to each of the interfaces, and a firewall services component coupled to the switching process component, said method comprising:

a) providing a session manager in the firewall services component;

b) providing a firewall module in the switching component;

c) instantiating a session, by said session manager, for data transfers within the firewall device, said sessions having header and payload information related to data transfers within the firewall device; and

d) instantiating a mini-session, by said session manager, corresponding to said instantiated session, said mini-session having header information related to data transfers within the firewall device, wherein said mini-session comprises instantiated software modules residing in the same address space as said switching process component.

18. The program storage device of claim **17**, said method further comprising:

a) intercepting data packets having a header and a payload component, by said firewall module, before reception by the packet filtering component; and

b) setting a "pass" flag in the intercepted data packets when said header component is the intercepted data packets matches said header information in said mini-session.

19. The program storage device of claim **17**, said method further comprising:

a) checking data packets for a "pass" flag, by said packet filtering component; and

b) bypassing "access control list" check, if a "pass" flag is found in said checked data packets.

20. The program storage device of claim **17**, said method further comprising:

a) intercepting data packets having a header and a payload component, by said firewall module, before reception by the packet filtering component; and

b) setting a "do not divert" flag in the intercepted data packets when said intercepted data packets packet inspection does not require application-level inspection.

21. The program storage device of claim **17**, said method further comprising:

a) checking data packets for a "do not divert" flag, by said firewall module; and

b) bypassing "access control list" check, if a "do not divert" flag is found in said checked data packets.

22. The program storage device of claim **17**, said method further comprising bypassing authorization with the firewall services component, by the firewall module, for data packets header information matching header information in said mini-sessions.

23. The program storage device of claim **17**, said method further comprising deleting said session and associated mini-session when data transfer associated with said sessions and mini-session is completed.

24. The program storage device of claim **17**, said method further comprising said session and associated mini-session when data transfer associated with said sessions and mini-session is idle past a predetermined timeout period.

25. The program storage device of claim **17**, said method further comprising updating context of said mini-session, by said firewall module, without sending packets to said firewall services component.

* * * * *

US006550012B1

(12) **United States Patent**
Villa et al.

(10) **Patent No.: US 6,550,012 B1**
(45) **Date of Patent: Apr. 15, 2003**

(54) **ACTIVE FIREWALL SYSTEM AND METHODOLOGY**

(75) Inventors: **Emilio Villa**, Ben Lomond, CA (US); **Adrian Zidaritz**, Danville, CA (US); **Michael David Varga**, Santa Clara, CA (US); **Gerhard Eschelbeck**, Peuerbach (AT); **Michael Kevin Jones**, Sunnyvale, CA (US); **Mark James McArdle**, San Carlos, CA (US)

(73) Assignee: **Network Associates, Inc.**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/328,177**

(22) Filed: **Jun. 8, 1999**

Related U.S. Application Data

(60) Provisional application No. 60/111,870, filed on Dec. 11, 1998.

(51) **Int. Cl.**^{7} **G06F 11/30**

(52) **U.S. Cl.** **713/201**; 713/200; 713/168; 713/170

(58) **Field of Search** 713/201, 200, 713/168, 170

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,790,790 A	*	8/1998	Smith et al.	709/206
5,802,178 A	*	9/1998	Holden et al.	713/151
5,815,657 A	*	9/1998	Williams et al.	713/200
5,828,832 A	*	10/1998	Holden et al.	713/201

* cited by examiner

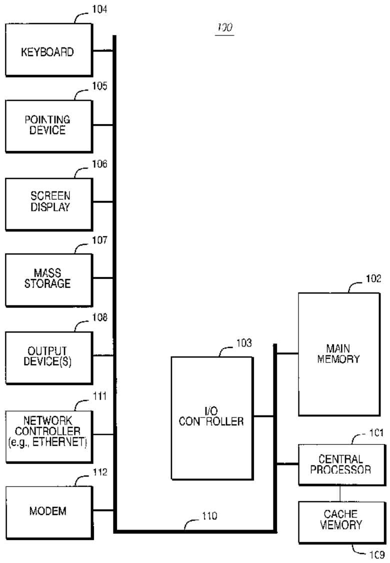
Primary Examiner—Thomas R. Peeso

(74) *Attorney, Agent, or Firm*—Patrick J. S. Inouye; Christopher J. Hamaty

(57) **ABSTRACT**

System and methodology providing automated or "proactive" network security ("active" firewall) are described. The system implements methodology for verifying or authenticating communications, especially between network security components thereby allowing those components to share information. In one embodiment, a system implementing an active firewall is provided which includes methodology for verifying or authenticating communications between network components (e.g., sensor(s), arbiter, and actor(s)), using cryptographic keys or digital certificates. Certificates may be used to digitally sign a message or file and, in a complementary manner, to verify a digital signature. At the outset, particular software components that may participate in authenticated communication are specified, including creating a digital certificate for each such software component. Upon detection by a sensor that an event of interest that has occurred in the computer network system, the system may initiate authenticated communication between the sensor component and a central arbiter (e.g., "event orchestrator") component, so that the sensor may report the event to the arbiter or "brain." Thereafter, the arbiter (if it chooses to act on that information) initiates authenticated communication between itself and a third software component, an "actor" component (e.g., "firewall"). The arbiter may indicate to the actor how it should handle the event. The actor or firewall, upon receiving the information, may now undertake appropriate action, such as dynamically creating or modifying rules for appropriately handling the event, or it may choose to simply ignore the information.

60 Claims, 8 Drawing Sheets



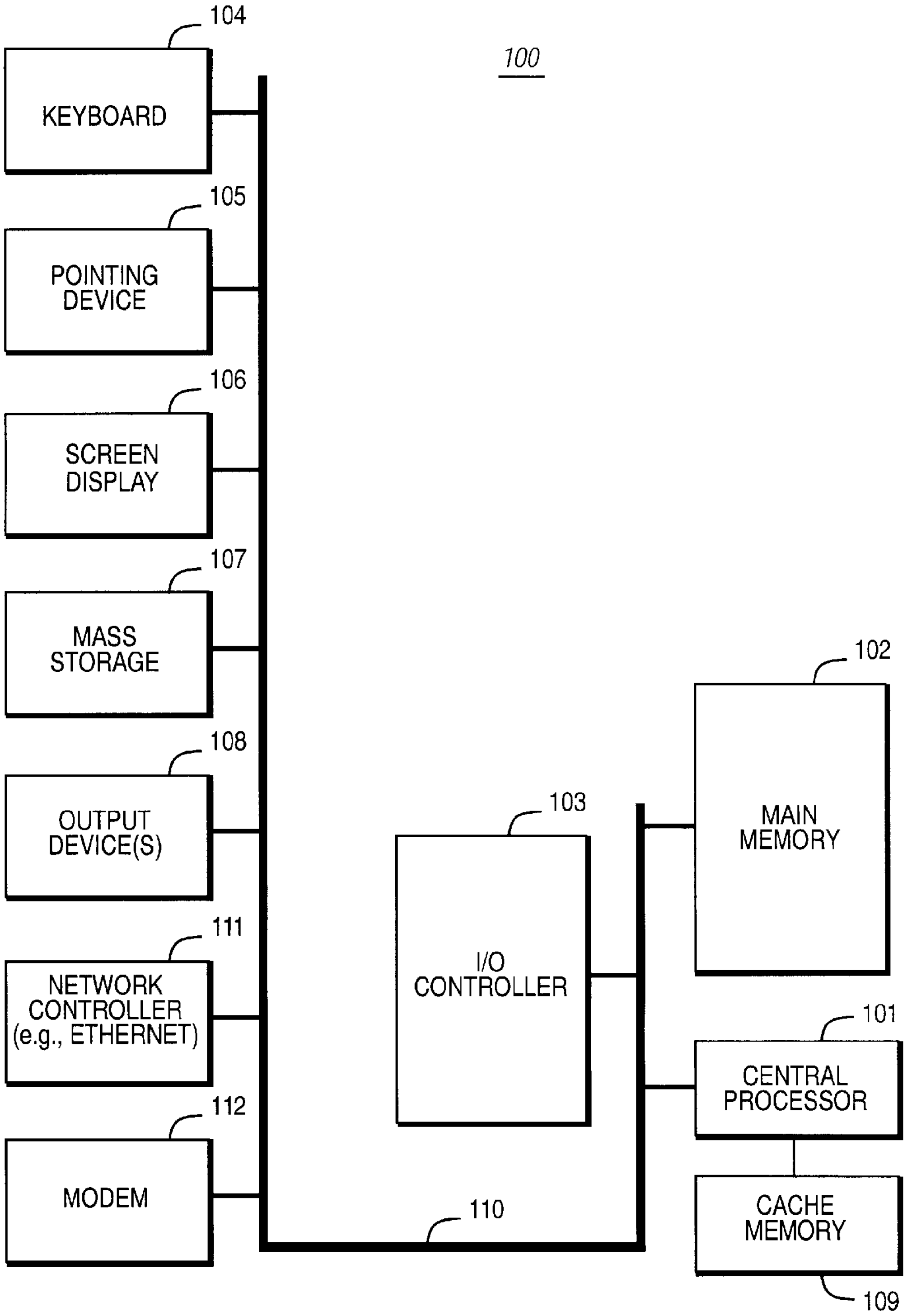

100
104
KEYBOARD
105
POINTING DEVICE
106
SCREEN DISPLAY
107
MASS STORAGE
108
OUTPUT DEVICE(S)
111
NETWORK CONTROLLER (e.g., ETHERNET)
112
MODEM
110
103
I/O CONTROLLER
102
MAIN MEMORY
101
CENTRAL PROCESSOR
CACHE MEMORY
109

FIG. 1

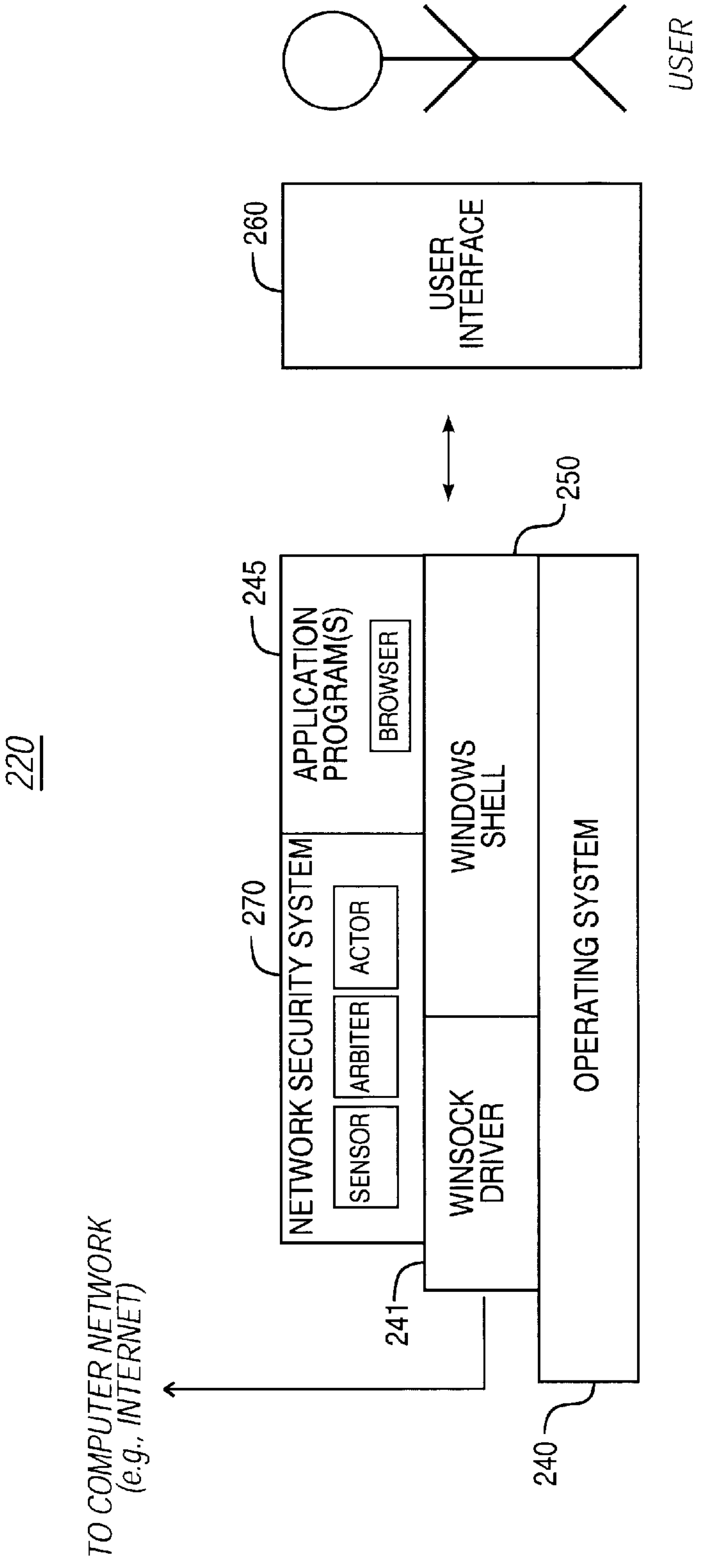

220
USER
260
USER INTERFACE
250
245
APPLICATION PROGRAM(S)
BROWSER
WINDOWS SHELL
270
NETWORK SECURITY SYSTEM
SENSOR
ARBITER
ACTOR
WINSOCK DRIVER
OPERATING SYSTEM
241
240
TO COMPUTER NETWORK (e.g., INTERNET)

FIG. 2

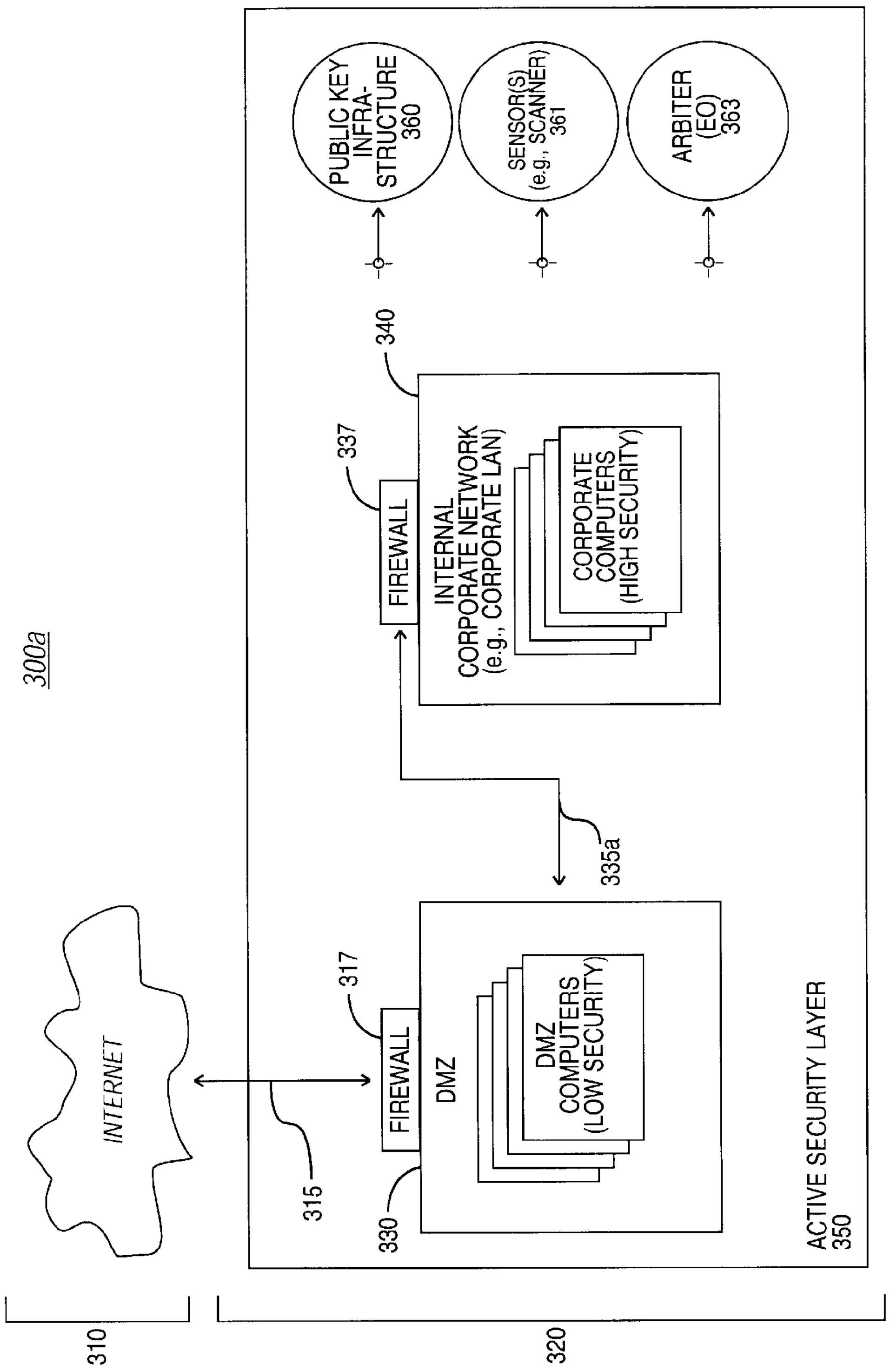

300a
INTERNET
315
310
FIREWALL
317
DMZ
DMZ COMPUTERS (LOW SECURITY)
330
335a
FIREWALL
337
INTERNAL CORPORATE NETWORK (e.g., CORPORATE LAN)
CORPORATE COMPUTERS (HIGH SECURITY)
340
PUBLIC KEY INFRA-STRUCTURE 360
SENSOR(S) (e.g., SCANNER) 361
ARBITER (EO) 363
ACTIVE SECURITY LAYER 350
320

FIG. 3A

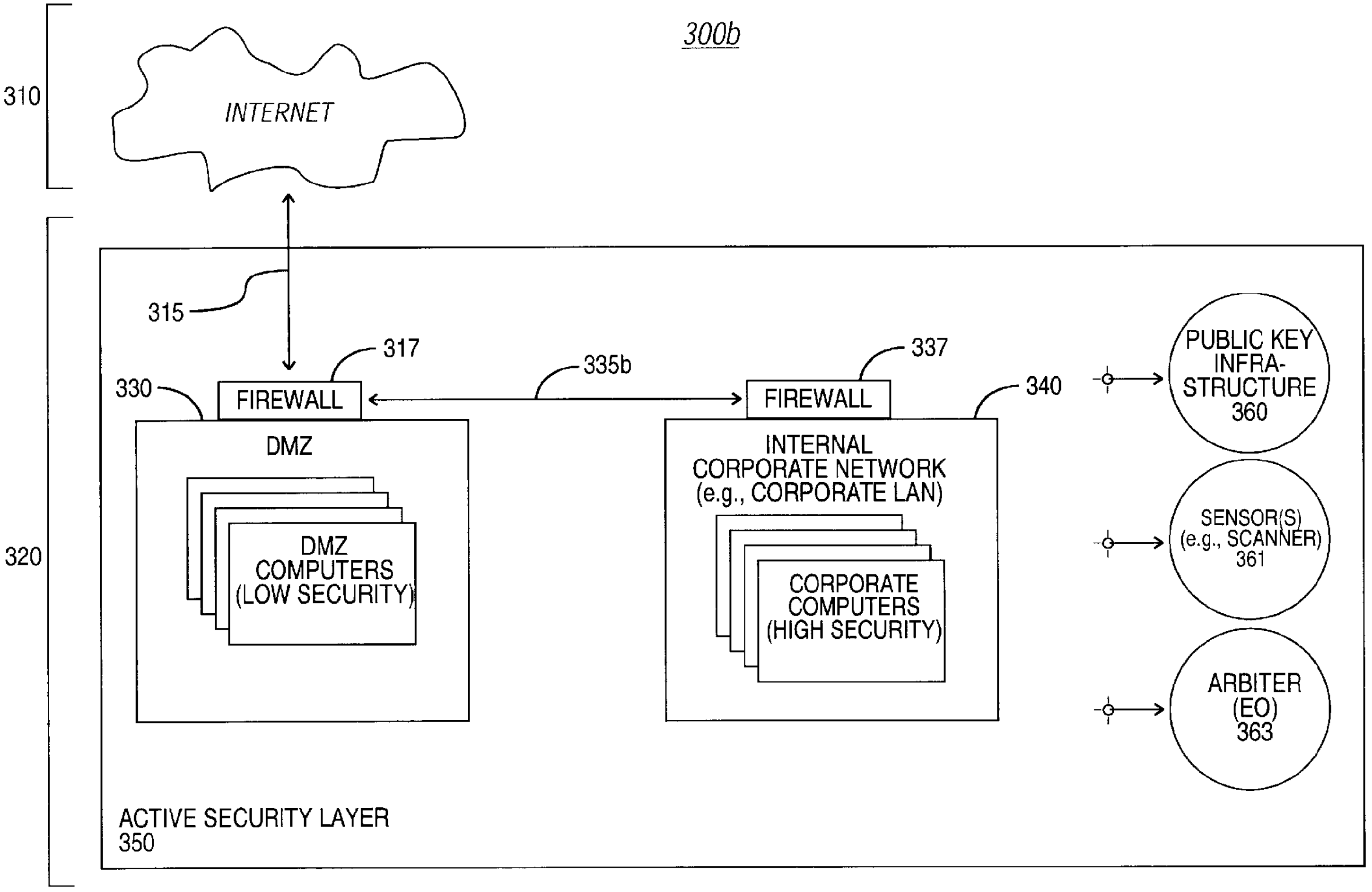

300b
310
INTERNET
320
315
317
FIREWALL
330
DMZ
DMZ
COMPUTERS
(LOW SECURITY)
335b
337
FIREWALL
340
INTERNAL
CORPORATE NETWORK
(e.g., CORPORATE LAN)
CORPORATE
COMPUTERS
(HIGH SECURITY)
PUBLIC KEY
INFRA-
STRUCTURE
360
SENSOR(S)
(e.g., SCANNER)
361
ARBITER
(EO)
363
ACTIVE SECURITY LAYER
350

FIG. 3B

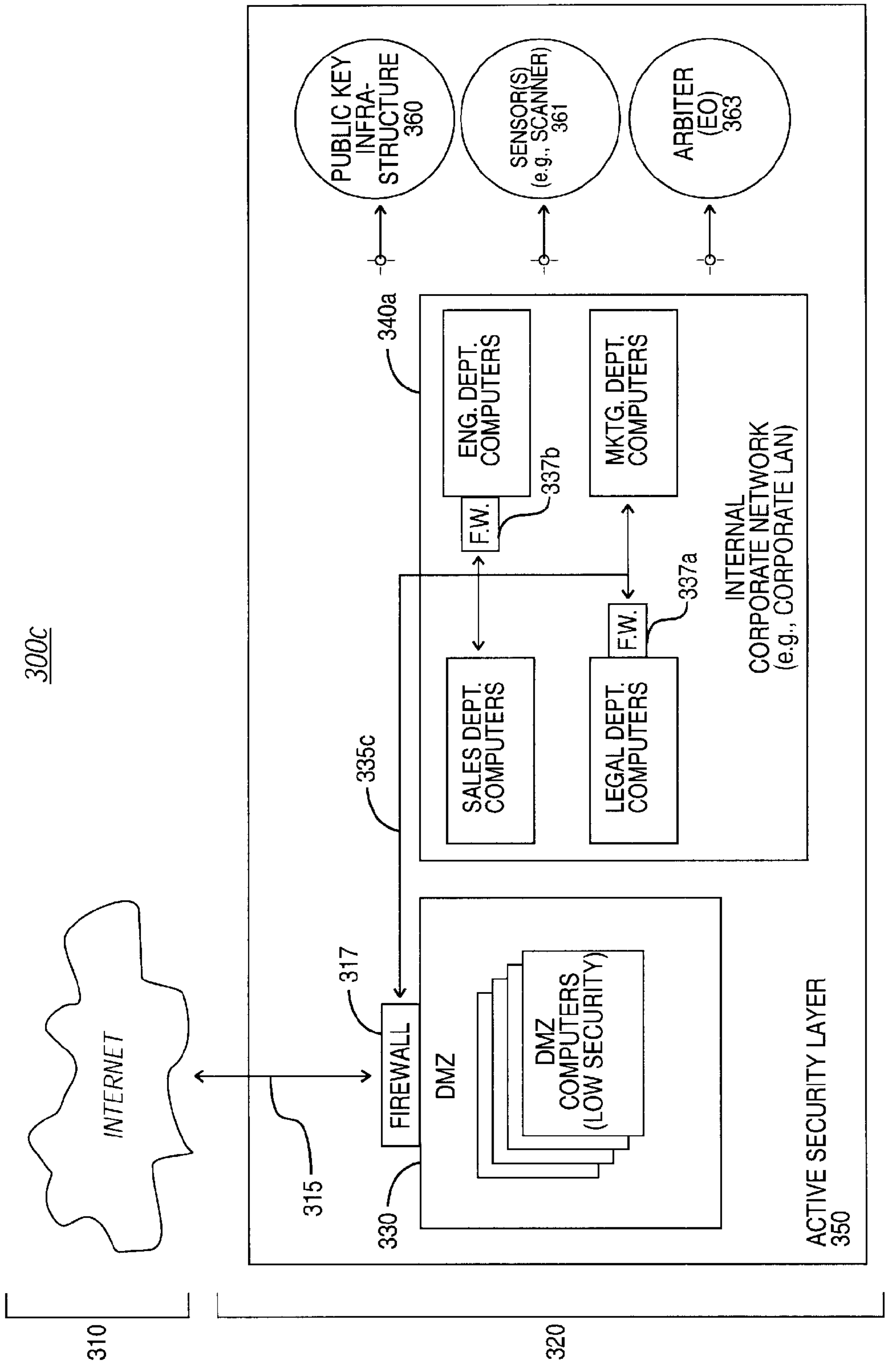

300c
INTERNET
315
FIREWALL
317
DMZ
330
DMZ COMPUTERS (LOW SECURITY)
335c
SALES DEPT. COMPUTERS
LEGAL DEPT. COMPUTERS
F.W.
337a
ENG. DEPT. COMPUTERS
F.W.
337b
MKTG. DEPT. COMPUTERS
340a
INTERNAL CORPORATE NETWORK (e.g., CORPORATE LAN)
PUBLIC KEY INFRA-STRUCTURE 360
SENSOR(S) (e.g., SCANNER) 361
ARBITER (EO) 363
ACTIVE SECURITY LAYER 350
310
320

FIG. 3C

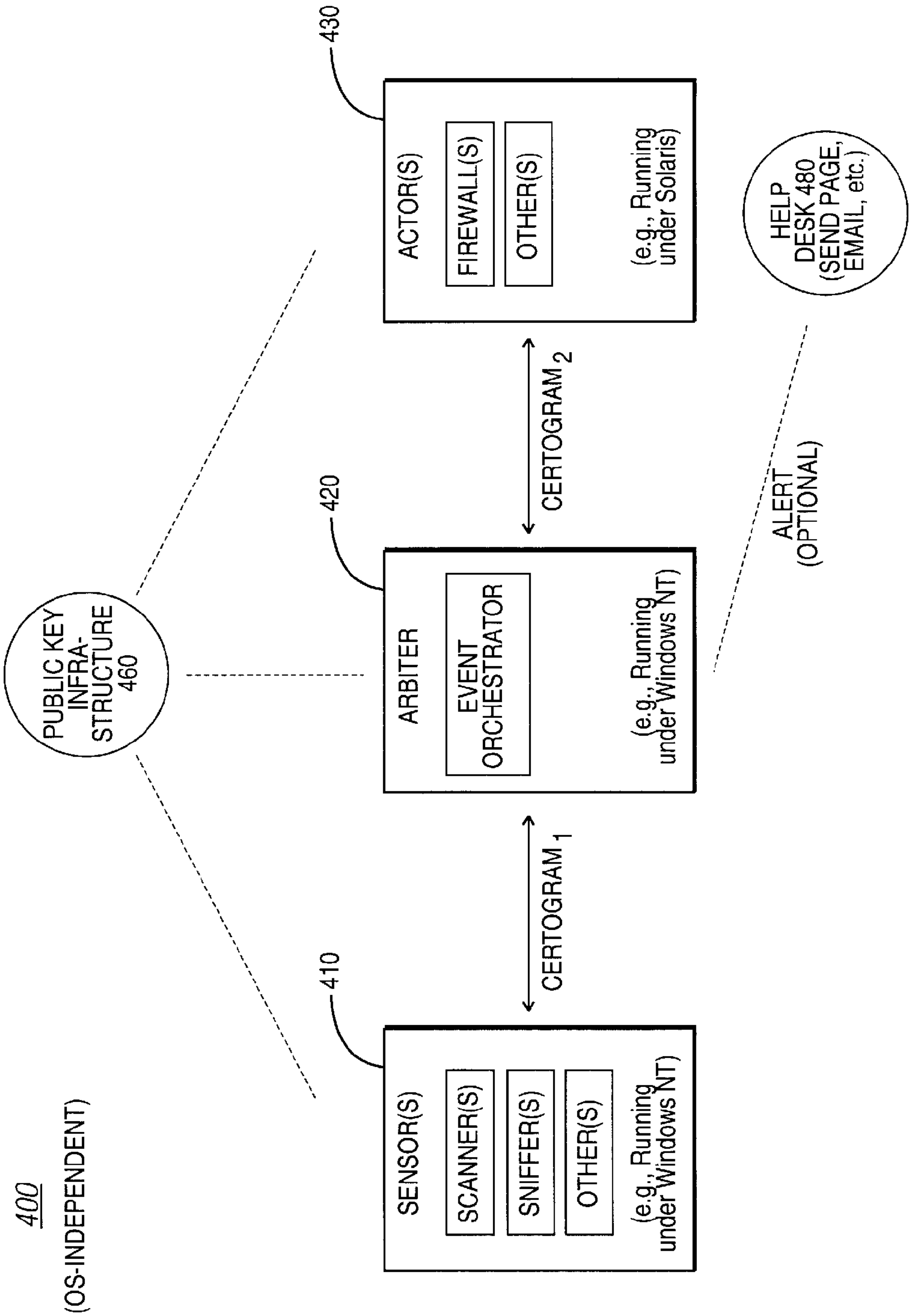

400
(OS-INDEPENDENT)
PUBLIC KEY INFRA-STRUCTURE 460
410
SENSOR(S)
SCANNER(S)
SNIFFER(S)
OTHER(S)
(e.g., Running under Windows NT)
CERTOGRAM$_1$
420
ARBITER
EVENT ORCHESTRATOR
(e.g., Running under Windows NT)
CERTOGRAM$_2$
430
ACTOR(S)
FIREWALL(S)
OTHER(S)
(e.g., Running under Solaris)
ALERT (OPTIONAL)
HELP DESK 480 (SEND PAGE, EMAIL, etc.)

FIG. 4

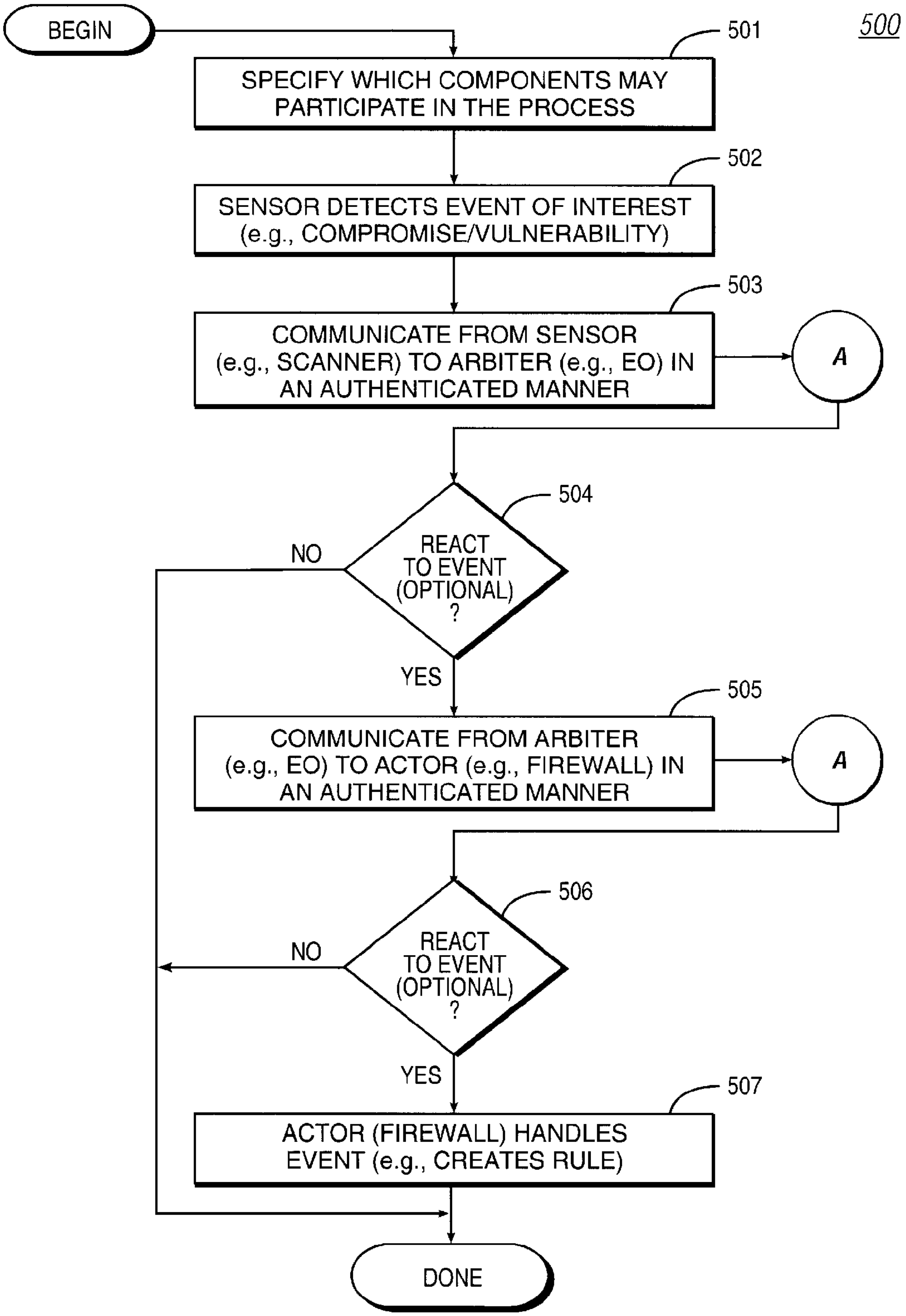

500
BEGIN
501
SPECIFY WHICH COMPONENTS MAY PARTICIPATE IN THE PROCESS
502
SENSOR DETECTS EVENT OF INTEREST (e.g., COMPROMISE/VULNERABILITY)
503
COMMUNICATE FROM SENSOR (e.g., SCANNER) TO ARBITER (e.g., EO) IN AN AUTHENTICATED MANNER
A
504
REACT TO EVENT (OPTIONAL) ?
NO
YES
505
COMMUNICATE FROM ARBITER (e.g., EO) TO ACTOR (e.g., FIREWALL) IN AN AUTHENTICATED MANNER
A
506
REACT TO EVENT (OPTIONAL) ?
NO
YES
507
ACTOR (FIREWALL) HANDLES EVENT (e.g., CREATES RULE)
DONE

FIG. 5A

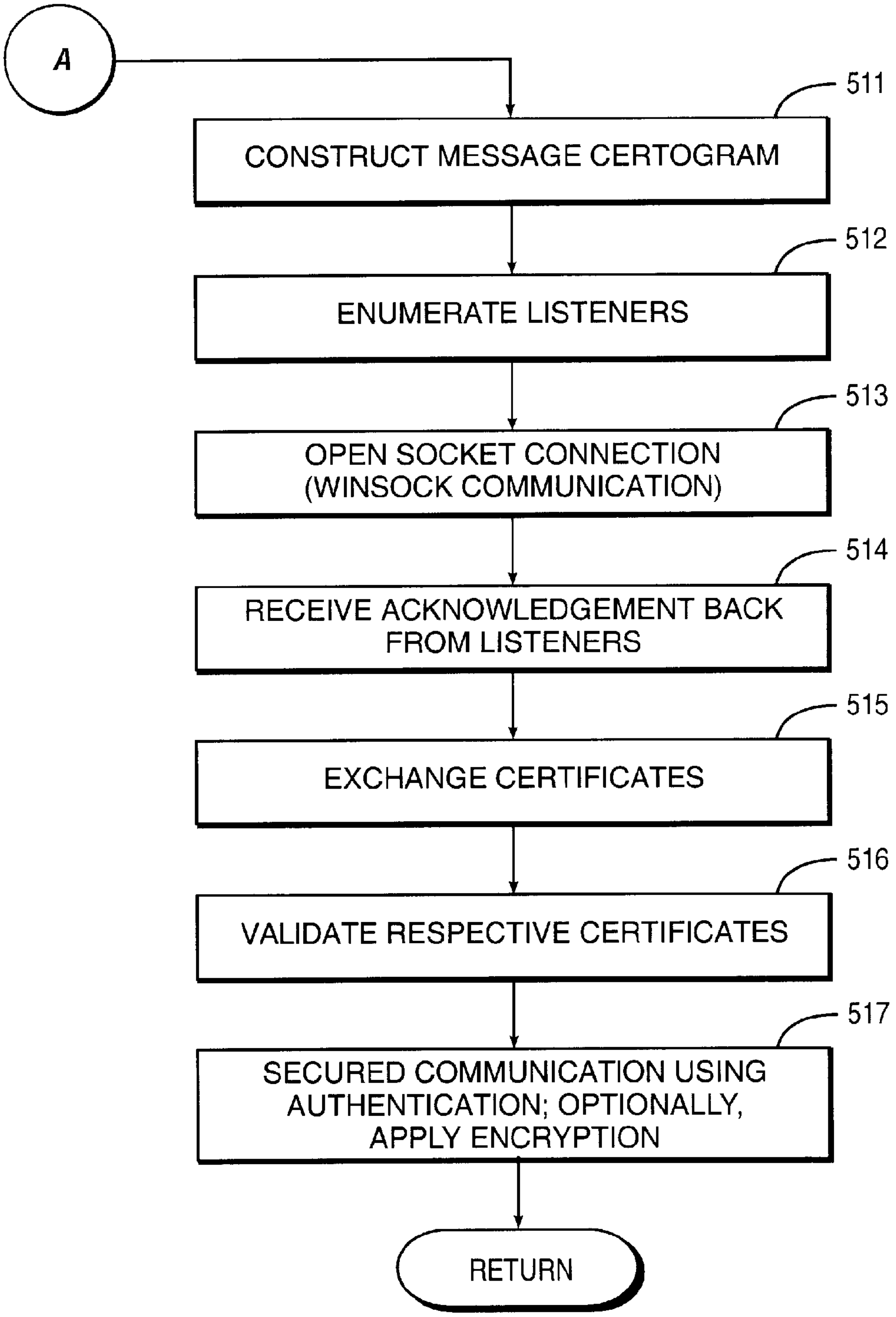

A
511
CONSTRUCT MESSAGE CERTOGRAM
512
ENUMERATE LISTENERS
513
OPEN SOCKET CONNECTION (WINSOCK COMMUNICATION)
514
RECEIVE ACKNOWLEDGEMENT BACK FROM LISTENERS
515
EXCHANGE CERTIFICATES
516
VALIDATE RESPECTIVE CERTIFICATES
517
SECURED COMMUNICATION USING AUTHENTICATION; OPTIONALLY, APPLY ENCRYPTION
RETURN

FIG. 5B

ACTIVE FIREWALL SYSTEM AND METHODOLOGY

RELATED APPLICATIONS

The present application claims the benefit of priority from and is related to the following commonly-owned U.S. provisional application: application Ser. No. 60/111,870, filed Dec. 11, 1998. The disclosure of the foregoing application is hereby incorporated by reference in its entirety, including any appendices or attachments thereof, for all purposes.

COPYRIGHT NOTICE

BACKGROUND OF THE INVENTION

The present invention relates generally to computer networks and, more particularly, to system and methods for facilitating the detection of events occurring in a computer network system (e.g., detection of vulnerability) and secure communication of such events within the system, as well as automated responses to such events.

The first personal computers were largely stand-alone units with no direct connection to other computers or computer networks. Data exchanges between computers were mainly accomplished by exchanging magnetic or optical media such as floppy disks. Over time, more and more computers were connected to each other using Local Area Networks or "LANs." In both cases, maintaining security and controlling what information a user of a personal computer can access was relatively simple because the overall computing environment was limited and clearly defined.

With the ever-increasing popularity of the Internet, particularly the World Wide Web ("Web") portion of the Internet, however, more and more personal computers are connected to larger networks. Providing access to vast stores of information, the Internet is typically accessed by users through Web "browsers" (e.g., Microsoft Internet Explorer™ or Netscape Navigator™) or other "Internet applications." Browsers and other Internet applications include the ability to access a URL (Universal Resource Locator) or "Web" site. The explosive growth of the Internet had a dramatic effect on the LANs of many businesses and other organizations. More and more employees need direct access through their corporate LAN to the Internet in order to facilitate research, competitive analysis, communication between branch offices, and send e-mail, to name just a few.

As a result, corporate IT (Information Technology) departments now face unprecedented challenges. Specifically, such departments, which have to date operated largely in a clearly defined and friendly environment, are now confronted with a far more complicated and hostile situation. As more and more computers are now connected to the Internet, either directly (e.g., over a dial-up connection with an Internet Service Provider or "ISP") or through a gateway between a LAN and the Internet, a whole new set of challenges face LAN administrators and individual users alike: these previously-closed computing environments are now opened to a worldwide network of computer systems. In particular, systems today are vulnerable to attacks by practically any perpetrators (hackers) having access to the Internet.

The general problem facing network environments is that security coverage/protection is generally not available 24 hours a day, seven days a week, at least not without great cost. Nevertheless, corporate networks are typically kept running at all times for various reasons, such as for hosting Web sites, FTP (File Transfer Protocol) sites, and the like. Although it is generally impractical to keep an IT team around 24 hours a day, seven days a week, corporate networks remain under constant threat of "attack," from both inside and outside sources.

There are several potential sources of attack. For example, an "inside" attack may occur as a result of the unauthorized act of an employee setting up a bogus FTP site, such as one containing confidential information that is not protected from access from outside the company. Another example of an inside attack is the unauthorized act of setting up a mail server (e.g., SMTP server) inside a corporate network, for sending unauthorized e-mail (e.g., completely bypassing company safeguards). "Outside" attacks typically occur as a result of unauthorized access to one's network by an outside perpetrator, that is, one existing outside the corporate "firewall." A typical example of such an attack would include unauthorized access to a valid FTP site which has accidentally been configured to have "writeable" directories which are not known to exist.

Firewalls are applications that intercept the data traffic at the gateway to a wide area network (WAN) and try to check the data packets (i.e., Internet Protocol packets or "IP packets") being exchanged for suspicious or unwanted activities. Initially firewalls have been used primarily to keep intruders from the LAN by filtering data packets. More recently, the concept has been expanded to include "proxy-based" firewall protection. A proxy-based firewall is one in which all relevant protocols are handled by an individual proxy, positioned (conceptually) between the incoming network card and the outgoing network card. In this manner, the proxy-based firewall can receive a connection from one side (e.g., incoming side) and apply relevant security checks before re-opening a corresponding connection on the other side (e.g., outgoing side).

Even with the availability of firewall technology, present-day techniques for detecting system compromise and vulnerabilities have occurred in a fairly non-automated fashion. Typically, an IT team routinely scans a company's network using scanning software, reviews a report of vulnerabilities, and then decides what firewall rules, if any, should be written. A particular problem with this approach is that existing firewalls have not, to date, served as an adequate substitute for IT personnel themselves. This stems from the fact that existing firewalls are simply static in nature and, thus, are unable to participate in a proactive, or even reactive, manner. When a breach in the network security or attack occurs, a firewall can only correctly handle the event if it has been programmed beforehand (e.g., by a system administrator) with a rule appropriate for the event. Since a firewall essentially serves as a repository of static rules, its functionality is limited by the ability of its system administrator to anticipate events and create rules for handling those events.

Often, however, an event will occur for which there is no rule. Since firewall rules themselves are not proactive, the firewall itself is unable to appropriately handle the event. Thus, events often require human intervention for appropriate handling. As these attacks can happen quite rapidly, such manual human intervention is itself often inadequate. Frequently, by the time IT personnel has detected an attack, it is too late: the damage (e.g., unauthorized access to confidential information) has already been done.

What is needed is a system with methodology that provides proactive protection for computer networks, thereby eliminating the need for continual, manual supervision and intervention for securing one's corporate network. Moreover, the underlying security and integrity of the proactive system itself should be assured, including communications within the system, so that the system itself does not introduce vulnerability to the network. In this manner, such a system may be employed to free IT personnel from the task of having to search for, and appropriately handle, system compromises in a non-automated manner. The present invention fulfills this and other needs.

SUMMARY OF THE INVENTION

System and methodology providing automated or "proactive" network security ("active" firewall) are described. In one embodiment, a system implementing an active firewall is provided which includes methodology for verifying or authenticating communications between network components (e.g., sensor(s), arbiter(s), and actor(s)), using cryptographic keys or digital certificates. Certificates may be used to digitally sign a message or file and, in a complementary manner, to verify a digital signature. These "digital signatures" allow authentication of messages, such that forgery of a signed message is not computationally feasible.

A methodology of the present invention for providing an active firewall may be summarized as follows. At the outset, particular software components that may participate in authenticated communication are specified, including creating a digital certificate for each such software component. The system has been configured by a system administrator for specifying which components of the system may participate in the process. Next, an event of interest occurs in the system, such as detection of a compromise or vulnerability in the system by a sensor (e.g., scanner). Upon occurrence of the event, the sensor component communicates information about the event in a secure, authenticated manner with another "listener" component, an arbiter or Event Orchestrator (EO).

Because of the system's existing configuration, the system already stores in its repository signed digital certificates (i.e., signed by the system administrator) for the two components, so that the components can proceed to engage in a digital conversation (i.e., communication session). Here, the sensor—acting as a "sender"—invokes the following substeps for effecting authenticated communication with one or more listeners. First, the sender creates a "certogram"—that is, a packet of information describing the event which is organized into a format suitable for transmission. In the currently-preferred embodiment, a certogram may be constructed using attribute/value pairs in plain text, such as <attribute>=<value>, with a delimiter employed for separating one pair from the next. The sender determines which component(s) are its listeners. This determination is performed by simply enumerating those component(s) that have been specified in the system configuration to be listeners for this component (e.g., sensor). The components in the system may be specified by an IP (Internet Protocol) address or other identification scheme.

Now, a socket connection may be established. In the currently-preferred embodiment, this is performed through PGP™ TLS using a sequence of API (application programming interface) calls into the PGPsdk™ run-time library. Here, the component opens a socket connection (communication to a particular IP address on a particular port), binds that to a session, and then broadcasts a message to that port announcing its presence. At this point in the process, the communication socket is simply a conventional stream socket; communication is not yet authenticated. If a listener is present, that listener will respond with an acknowledgment accepting the socket connection. An acknowledgment may be received back from one or more listeners. Now that the communication layer is established, the method may proceed to the next substep, for exchanging certificates with the listener(s).

The respective sender/listener(s) components each validate the certificate received from the other. Validation may proceed in a conventional manner, for example using X.509 validation, including determining the level of trust and validity (including, for instance, the expiration, revocation, and disablement) of a given certificate. If each respective component is able to successfully validate the certificate received from the other, secure communication ensues. From that point on, communication occurs in a secure, authenticated manner, with each message or blob being digitally signed or fingerprinted, for instance, using a cryptographic hash or message digest. Any alteration to the message breaks the digital fingerprint and, therefore, may easily be detected. If desired, encryption may also be (optionally) applied to the communication messages. In those embodiments intended for export from the United States, however, encryption may be limited (e.g., as to key length) or removed.

Upon return back to the main or controlling method (i.e., after completion of the foregoing substeps), the listener(s) decides whether to act on or pass on the event reported by the certogram, or simply to ignore it. If the event is not to be acted on or passed on, the method is done. Typically, however, the reported event maps to a script-defined event handler in the listener (e.g., Event Orchestrator or EO) which, in turn, desires to notify yet another listener, the actor (e.g., firewall). Communication may therefore continue with the arbiter (EO) communicating with the target actor(s) in an authenticated manner, as above, with the arbiter (EO) as the sender and the actor (firewall) as the listener. The actor or firewall, upon receiving the certogram, may now undertake appropriate action, such as dynamically creating or modifying rules for appropriately handling the event, or it may choose to ignore the event (e.g., if the event is a duplicate of a previous event or if the event is covered by (or is a sub-set of) an existing firewall rule).

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a high-level block diagram illustrating a computer in which the present invention may be embodied.

FIG. 2 is a high-level block diagram illustrating a software system for controlling the operation of the computer of FIG. 1, including sensor(s), arbiter(s), and actor(s) network security components.

FIG. 3A is a high-level block diagram illustrating a computer network system in which the present invention is preferably embodied.

FIG. 3B is a high-level block diagram illustrating a computer network system in which the present invention is preferably embodied, having a network configuration in which a primary firewall (e.g., for DMZ computers) communicates directly with a secondary firewall (e.g., for corporate (internal) computers).

FIG. 3C is a high-level block diagram illustrating a computer network system in which the present invention is preferably embodied, having a network configuration in which a primary firewall (e.g., for DMZ computers) communicates directly with a multitude of secondary firewalls (e.g., for individual department-level corporate (internal) computers).

FIG. 4 is a high-level block diagram illustrating an operating system-independent network security system implementing an "active firewall" of the present invention.

FIGS. 5A–B comprise a flowchart illustrating a method of the present invention for providing "active firewall" protection for a computer network, which includes authenticated communication between software components.

GLOSSARY

The following definitions are offered for purposes of illustration, not limitation.

Actor: A device or computer-implemented process that actively responds to a particular situation, such as a "firewall" or "router." A firewall may, for instance, actively close a particular port in response to a given situation.

Arbiter: A device or computer-implemented process that serves as a central "brain" for coordinating events which occur in a system (e.g., network). For instance, an "Event Orchestrator" (EO) may be employed as a system's arbiter for handling events which are detected in a system.

Authentication: The determination of the origin of encrypted information through the verification of someone's digital signature or someone's public key by checking its unique fingerprint.

Certogram: A blob or chunk of data comprising value/attribute pairs, which may be employed for transmitting information about an event detected in a network system.

Certify: To sign another person's public key.

Certifying Authority: One or more trusted individuals are assigned the responsibility of certifying the origin of keys and adding them to a common database.

Decryption: A method of unscrambling encrypted information so that it becomes legible again. The recipient's private key is used for decryption.

Digital Signature: See signature.

Encryption: A method of scrambling information to render it unreadable to anyone except the intended recipient, who must decrypt it to read it.

Key: A digital code used to encrypt, sign, decrypt and verify e-mail messages and files. Keys come in key pairs and are stored on keyrings.

Key Escrow: A practice where a user of a public key encryption system surrenders their private key to a third party thus permitting them to monitor encrypted communications.

Key Fingerprint: A uniquely identifying string of numbers and characters used to authenticate public keys. For example, one can telephone the owner of a public key and have him or her read the fingerprint associated with their key so one can compare it with the fingerprint on one's copy of their public key to see if they match. If the fingerprint does not match, then one knows one has a bogus key.

Key ID: A legible code that uniquely identifies a key pair. Two key pairs may have the same User ID, but they will have different Key IDs.

Key Pair: A public key and its complimentary private key. In public-key cryptosystems, like the PGP™ program, each user has at least one key pair.

Keyring: A set of keys. Each user has two types of keyrings: a private keyring and a public keyring.

Message Digest: A compact "distillate" of one's message or file checksum. It represents one's message, such that if the message were altered in any way, a different message digest would be computed from it.

Passphrase: A series of keystrokes that allow exclusive access to one's private key which one uses to sign and decrypt e-mail messages and file attachments.

Plaintext: Normal, legible, unencrypted, unsigned text.

Private Key: The secret portion of a key pair used to sign and decrypt information. A user's private key should be kept secret, known only to the user.

Private Keyring: A set of one or more private keys, all of which belong to the owner of the private keyring.

Public Key: One of two keys in a key pair used to encrypt information and verify signatures. A user's public key can be widely disseminated to colleagues or strangers. Knowing a person's public key does not help anyone discover the corresponding private key.

Public Keyring: A set of public keys. One's public keyring includes one's own public key(s).

Public-Key Cryptography: Cryptography in which a public and private key pair is used, and no security is needed in the channel itself.

Sign: To apply a signature.

Signature: A digital code created with a private key. Signatures allow authentication of information by the process of signature verification. When one signs a message or file, the PGP™ program uses one's private key to create a digital code that is unique to both the contents of the message and one's private key. Anyone can use one's public key to verify one's signature.

Sensor: Any type of device or computer-implemented process for gathering information about a network. Examples of sensor-type software include scanners (including anti-virus scanners), monitors (i.e., software which "listens" for network intrusions), sniffers, or the like.

Text: Standard, printable text (e.g., 7-bit ASCII text).

Trusted: A public key is said to be trusted by the user if it has been certified by the user or by someone the user has designated as an introducer.

User ID: A text phrase that identifies a key pair. For example, one common format for a User ID is the owner's name and e-mail address. The User ID helps users (both the owner and colleagues) identify the owner of the key pair.

Verification: The act of comparing a signature created with a private key to its public key. Verification proves that the information was actually sent by the signer, and that the message has not been subsequently altered by anyone else.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

The following description will focus on the presently-preferred embodiment of the present invention, which is operative in an Internet-connected environment, including, for instance, client machines running under a client operating system (e.g., the Microsoft® Windows 9x environment) and connected to a network running under a network operating system (e.g., the Microsoft® Windows NT or Windows 2000 environment), with connectivity to an open network such as the Internet. The present invention, however, is not limited to any particular one application or any particular environment. Instead, those skilled in the art will find that the system and methods of the present invention may be advantageously applied to a variety of system and application software, including database management systems, word processors, spreadsheets, and the like, operating on a variety of different platforms, including Macintosh®, UNIX®, NextStep®, Linux™, and the like. Therefore, the description of the exemplary embodiments which follows is for purposes of illustration and not limitation.

Computer System Architecture

A. Hardware for listeners (e.g., firewall) and senders (e.g., scanner)

The invention is generally embodied on a computer system including one or more computer systems, such as computer system **100** of FIG. **1**, operating on a network. System **100** comprises a central processor **101**, a main memory **102**, an input/output controller **103**, a keyboard **104**, a pointing device **105** (e.g., mouse, track ball, pen device, or the like), a display or screen device **106**, and a mass storage **107** (e.g., hard or fixed disk, removable floppy disk, optical disk, magneto-optical disk, or flash memory), a network controller or interface card **111** (e.g., Ethernet), and a modem **112** (e.g., 28.8 K baud modem or ISDN modem). Although not shown separately, a real-time system clock is included with the system **100**, in a conventional manner. Processor **101** includes or is coupled to a cache memory **109** for storing frequently accessed information; memory **109** may be an on-chip cache or external cache (as shown). One or more input/output (I/O) controllers(s) or device(s) **108** are included in the system **100**, as desired. I/O devices **108** may include, for instance, a laser printer, such as an HP LaserJet® printer available from Hewlett-Packard of Palo Alto, Calif. As shown, the various components of the system **100** communicate through a system bus **110** or similar architecture. The system itself communicates with other systems via the network interface card (NIC) **111** (e.g., available from 3Com of Santa Clara, Calif.) and/or modem **112** (e.g., also available from 3Com), connected to a network (e.g., Ethernet network). In a preferred embodiment, the system **100** includes an IBM PC-compatible personal computer, available from a variety of vendors (including IBM of Armonk, N.Y.).

B. System software

Illustrated in FIG. **2**, a computer software system **220** is provided for directing the operation of the computer system **100**. Software system **220**, which is stored in system memory **102** and on storage (e.g., disk memory) **107**, includes a kernel or operating system (OS) **240** and a windows shell **250** (e.g., integrated into the OS or standalone). One or more application programs, such as client application software or "programs" **245** may be "loaded" (i.e., transferred from storage **107** into memory **102**) for execution by the system **100**. The software system **220** includes a communication layer or driver **241** (e.g., Microsoft Winsock) for effecting communication with a network, such as the Internet.

System **220** includes a user interface (UI) **260**, preferably a Graphical User Interface (GUI), for receiving user commands and data. These inputs, in turn, may be acted upon by the system **100** in accordance with instructions from OS module **240**, windows shell **250**, and/or client application module(s) **245**. The UI **260** also serves to display the results of operation from the OS **240**, windows shell **250**, and application(s) **245**, whereupon the user may supply additional inputs or terminate the session. OS **240**, windows shell **250**, and UI **260** can be provided by Microsoft® Windows 95/98, or by Microsoft® Windows NT/2000 (available from Microsoft Corporation of Redmond, Wash.). Alternatively, these can be provided by IBM OS/2 (available from IBM of Armonk, N.Y.), Macintosh OS (available from Apple Computers of Cupertino, Calif.), or Linux (available from a variety of vendors). Although shown conceptually as separate modules, windows shell **250** and UI **260** are typically provided by OS vendors as integral parts of the operating system (here, OS **240**).

As is also shown in FIG. **2**, the software system **220** includes a network security system **270** of the present invention. The network security system **270**, which includes sensor(s), arbiter(s), and actor(s) network security components in communication with one another through a secured communication layer, implements an active firewall architecture and methodology of the present invention for protecting one's network. Although the network security components are shown implemented in a single computer (i.e., system **100** operating under the control of software system **220**), the components may be implemented in a single computer system or multi-computer system, as desired. In a preferred embodiment, these components would typically reside on individual machines (e.g., each itself comprising a computer, such as system **100** operating under the control of software system **220**), connected to a computer network system (e.g., corporate network with Internet connectivity).

Construction and operation of the network security system **270**, in accordance with the present invention, will now be described in detail. Although the description which follows includes specific detail about network security components used in the preferred embodiment, the present invention itself does not require use of any vendor-specific components. Instead, the present invention may be implemented using sensor(s), arbiter(s), or actor(s) components other than those referenced below, including ones from third party vendors, as well as mixing and matching components from a variety of different vendors.

Active Firewall with Proactive Security Layer

A. Introduction

1. Eliminating Human-to-human Communication

An important design consideration in the creation of an active firewall of the present invention is the replacement of communication between humans with software-based communication. Consider, for instance, the following typical practice today. Suppose an IT worker running a scanner program discovers a vulnerability on a company network. If that individual does not have appropriate access level (i.e., system privileges) to make modifications for correcting the situation (as is often the case), he or she will have to inform the appropriate member of the IT team who is capable of making the appropriate changes. In this act of human-to-human communication, it is taken for granted that the communication is secure—that is, that there is no unauthorized participant or listener. In other words, the communication occurs in a manner such that "spoofing" is not possible. Thus, in this example, the IT worker achieves secure, authenticated communication by telling someone that he or she personally knows and trusts.

Automation of this level of communication in software exposes the network to potential problems, however. For instance, an unauthorized party may be listening in and may therefore also be privy to the newly-uncovered vulnerabilities discovered on the network. Thus, automation of communication potentially exposes the network to inappropriate dissemination of information (e.g., about network vulnerabilities) as well as allowing an unauthorized party to spoof the network (e.g., "man-in-the-middle" attack). Therefore, an important design consideration is that communication within the system is secure—that is, capable of authentication (e.g., using X.509 certificates). Although authentication itself does not require encryption, the communications may also be (optionally) encrypted, if desired, for thwarting an unauthorized listener.

2. General Applicability of Secured Intra-system Communication

Although the authenticated communication approach of the present invention is employed, in a preferred embodiment, for implementing proactive network security (i.e., active firewall), the approach has general applicability beyond such an implementation. Instead, the approach may be applied to any application where system-wide secure communication is desired. Thus, the present invention may be applied not only for managing network security but, also, any task which requires communication or coordination among components of a system where authentication of those components participating is desirable. For instance, the approach may be employed for automating system-critical functions of a system, updating of the system or extending system configuration information. In this manner, therefore, the communication approach of the present invention may be employed to allow different modules or pieces of software within a system (e.g., corporate network) to exchange information in an automated, yet secure, manner.

3. Cooperative Communication Model

In the currently-preferred embodiment, communication between software modules in the active firewall (e.g., certogram-based communication) is generally cooperative. In this model, a particular software component or module may undertake to inform another software module about a particular item or event. Whether the receiving software module actually decides to act on and/or pass on that information, however, would be at its own discretion (i.e., would be permissive, not mandatory). Accordingly, communication occurs in a manner similar to that of human communication: a software module offers information to other software modules but does not, generally, control (i.e., execute commands on) the other software modules. In this manner, communication occurs in a fashion which is not only asynchronous but is also "ignoreable."

B. Proactive network security system ("active firewall")

1. Corporate Network Configuration

In order to understand network architecture employed in the present invention (i.e., which includes multiple connected computers), it is helpful to consider an implementation employing a simple, yet common, network: an internal company network. FIG. 3A illustrates an overview of an embodiment of the present invention implemented in such an environment. At the highest level, network system **300***a* includes the Internet **310** (i.e., connectivity to the outside world through the Internet) and the "corporate network" **320** (i.e., the company's computing environment apart from the Internet). Given that most companies today must have a presence on the Internet, IT departments often employ the concept of a DMZ or "de-militarized zone," which exists between a company's own internal network and the Internet. Thus, since the company wants to have a presence on the Internet but not have that presence put the corporate network at risk, the corporate network **320** includes a DMZ **330**. In practice, DMZ **330** is implemented as a collection of server computers (e.g., collection of computer systems **100**) that are at very low risk for incoming connections. DMZ machines are relatively unsecured for incoming connections from the Internet, so that they are relatively accessible to the outside world. Since they are easily attacked, DMZ machines would typically not store confidential information.

In stark contrast to the foregoing relatively-unsecured connection with the Internet, the connection of the DMZ machines **330** to other computers within the corporate network—corporate computers **340**—is often a more secured connection, as the corporate computers **340** typically store sensitive company information. Accordingly, such an architecture would typically include an additional or secondary firewall **337** between the DMZ **330** and corporate computers **340**. Thus, the connection **315** between the Internet and DMZ computers **330** is relatively unsecured, while the connection **335***a* between the DMZ computers **330** and corporate computers **340** generally occurs in a more secure environment (e.g., one storing more sensitive content). The net effect is that communication both to and from the DMZ machines **330** to corporate computers **340** is more sensitive to the company. Often, however, at least some access restrictions are placed on the DMZ **330**. This is achieved by including a front-line or primary firewall **317** (e.g., proxy-based firewall), juxtaposed between the Internet and DMZ computers **330**. This will prevent, for example, an unauthorized party from posting its own HTML files on the DMZ computers **330**, for taking over the company's Web site.

The corporate network **320** is surrounded by an Active Security Layer **350** of the present invention. The layer **350**, which is connected to the various components of the corporate network **320**, supports authenticated communication among components within the system, through the use of digital certificates supported by a Public Key Infrastructure (PKI) **360**. Here, the PKI serves as a provider of certificates. When communication between system components is desired, the components exchange digital certificates (if not done so already) and thereafter transmit authenticated messages.

FIGS. 3B–C illustrate that the configuration of the firewalls of the network system can vary substantially, depending on the individual needs of the corporation. In FIG. 3B, the network system (now shown as system **300***b*) has been reconfigured such that the primary firewall **317** communicates directly with the secondary firewall **337** (e.g., also a proxy-based firewall), through direct connection **335***b*. Thus, as shown in this configuration, connection **335***b* does not pass through DMZ computers **330**. In FIG. 3C, the network system (now shown as system **300***c*) has been reconfigured such that the primary firewall **317** communicates directly with a plurality of secondary firewalls (e.g., secondary firewall **337***a*, **337***b*). As in the configuration for system **300***b*, the configuration of system **300***c* provides direct communication between the primary firewall (i.e., firewall **317**) and the secondary firewalls (e.g., firewalls **337***a*, **337***b*), that is, it does not pass through DMZ computers **330**.

Construction and operation of the present invention implemented in a network system will now be described, with particular emphasis on methodology of the present invention for providing an active firewall. Although the following description will focus on an exemplary firewall configuration, the present invention itself is independent of any particular firewall configuration and, instead, may be implemented in any one of a variety of firewall configurations (such as shown above).

2. "Active Firewall" Components/architecture

a. General Components

As shown in FIG. **4**, active firewall system **400** of the present invention includes sensor(s) **410**, arbiter(s) **420**, and actor(s) **430** components connected to a Public Key Infrastructure (PKI) **460**, which serves as a provider of certificates. If desired, the arbiter(s) **420** may optionally be connected to a Help Desk **480**, for example, for transmitting automated messages to human recipients (e.g., IT personnel). The following description focuses on a currently-preferred embodiment having a single arbiter **420**; those skilled in the art will appreciate, however, that the embodiment may be extended to include multiple arbiters, if desired, thereby providing failure-safety and load-balancing features.

As shown, communication between the components is in the form of the certograms, which are described in further detail below. In a preferred embodiment, each certogram has a finite existence. For example, certogram$_1$ may be used to communicate from a particular sensor **410** (e.g., scanner) to the arbiter **420**, but the arbiter **420** will not reuse certogram$_1$ to communicate with one of the actors **430** (e.g., firewall). Instead, the arbiter will create a new certogram$_2$—one specific for its communication with the (target) actor. The system itself is independent of any particular operating system, such that each component may run under the same or different operating system. The individual components themselves will now be described in further detail.

b. "Sensors"

Sensors **410** include any type of device or computer-implemented process that gathers information about a network, such as information about attacks or intrusions, vulnerabilities, network overload, and other similar conditions or events of interest. Examples of sensor-type software include scanners (including antivirus scanners), monitors (i.e., software which "listens" for network intrusions or attacks), sniffers (for detecting overload), or the like.

As a specific example, sensors **410** may include CyberCop™ Scanner (available from Network Associates, Inc. of Santa Clara, Calif.), for detecting network vulnerabilities. Scanner software proactively scans available computers/hosts on a network for detecting vulnerability. In operation, scanner software performs a comprehensive examination on each host on the network that is of interest (i.e., according to how the scanner is configured). On each machine, the scanner will run a series of tests, ranging from password grinding to trace routing, for instance. The series of tests themselves are configurable. Even though the scanner software is capable of running a wide range of tests, it itself is basically a mechanism for reporting network vulnerability. In particular, the scanner software itself generally includes only rudimentary capability for fixing vulnerabilities that it finds. As a result, its primary use is to provide reports to an IT worker who must then manually correct any system vulnerabilities, when that person is available. (As described below, a firewall is generally far better equipped to fix vulnerabilities.)

In the presently-preferred embodiment, sensors **410** include Network Associates' CyberCop™ Scanner (CSC) software. CyberCop™ Scanner v5.0 (WinNT) and CyberCop Scanner 2.5 (Linux) are available from Network Associates, Inc. of Santa Clara, Calif. (e.g., Part Nos./SKUs CSC-DRCT-NA-500-S1, CSC-DRCT-NA-500-S, and CSC-DRCT-NA-500-P), the documentation of which is hereby incorporated by reference. However, the present invention does not require use of scanner software but, instead, may employ other types of sensors, as desired.

c. "Arbiter" and "Actors"

The arbiter **420** may be implemented as Event Orchestrator-type (EO) software, such as "Event Orchestrator" (available from Network Associates, Inc. of Santa Clara, Calif.), to serve as a central "brain" for coordinating events which occur in a system (e.g., network). The EO is typically used, for example, to implement an electronic "Help Desk"—an automated system where users can post requests for computer-related help. In response to user-posted

requests, for example, the EO can automatically generate and send appropriate messages to responsible parties, including for instance sending e-mail messages, network messages, electronic pages, or the like to IT personnel.

EO software may be extended for coordinating or orchestrating the processing of problems occurring on a network, other than user-related ones. In particular, when an event occurs in the network system, it is reported to the central brain, EO, for appropriate handling. In accordance with the present invention, when an event of interest occurs in the network, the event is reported to the EO as a "certogram." Certogram refers to authenticated communication of a message within the system, using internal digital certificate-based, authenticated communication (i.e., a "certified electronic telegram"). For instance, when a scanner detects a vulnerability in the network, it may communicate this to the EO by creating a certogram that is then transported to the EO using the underlying Active Security Layer **350**, as illustrated in FIG. **3**. Upon receipt of a message reporting such an event, the EO maps the certogram for appropriate action. In response to a particular message, for example, the EO itself may generate an alert to the Help Desk **480** which, in turn, transmits a human-readable message, such as an e-mail message or pager message, or the EO may generate a machine-readable message, such as a network message to one or more "actors" (typically firewalls, such as firewall **337**). In either case, the EO will create a new certogram for effecting communication with its target listener.

Consider, as an example, the detection of an FTP writeable directory by a scanner. Upon receipt at the EO of a certogram reporting the problem, the EO may employ the Active Security Layer **350** to create a new certogram informing the actor (e.g., firewall) of detection of the network vulnerability and do so in an authenticated manner. Here, the network is configured at the outset (e.g., by a system administrator) to specify which components may participate in the secured communication process. For example, a system administrator may specify that a particular set of scanners (i.e., scanner software) may communicate with a particular EO. The actual configuration or registration of components is done in a manner somewhere to that used in a hierarchy ("web") of trust, where parties are introduced to one another by an "introducer."

In the presently-preferred embodiment, firewall **337** includes Network Associates' Gauntlet™ Firewall (GFX) software. Gauntlet™ Firewall (GFX), which includes Gauntlet Firewall v5.0 (Windows NT and Solaris), Gauntlet VPN 5.0 (Windows NT and Solaris), Net Tools PKI Server v1.0 (Windows NT, DSA and RSA+DSA versions), is available (per server basis) from Network Associates, Inc. of Santa Clara, Calif. in a variety of licensing configurations (e.g., Part Nos./SKUs, GFX-DRCT-NA-500-1V (1 year), GFX-DRCT-NA-500-SV (2 year), GFX-DRCT-NA-500-PV (perpetual)), the documentation of which is hereby incorporated by reference.

The firewall (actor), assured of the authenticity of the message, may create a new rule on-the-fly to address the vulnerability. Thus for the example of a writeable FTP directory, the firewall may create a rule specifically handling the vulnerability (e.g., by blocking access to a specific port or machine). Thus as demonstrated by the example, the system of the present invention supports authenticated digital certificate-based communication among the components within the network system, thus allowing those components to share information in a manner providing an active firewall or proactive network security system.

3. Cryptographic-secured System Communication

Authenticated communication placed on the underlying physical network (i.e., "wire") may be implemented using presently-available cryptographic code libraries, such as PGPsdk™ (version 1.6) available from Network Associates, Inc. of Santa Clara, Calif. In order to understand the application of such technology, it is helpful to briefly review cryptographic technology.

Generally, cryptographic systems use either "secret-key" encryption or "public key" encryption. In "secret-key" encryption, a single key is used for both encryption and decryption. Consider, for example, a user (sender) who wants to send an electronic mail or "e-mail" message to a colleague (recipient) in a secured manner, such that no one who intercepts the message will be able to read it. If the sender employs a cryptographic "secret key" to encrypt the message, the recipient, in turn, must also use the same key to decipher or decrypt the message. As a result, the same key must be initially transmitted via secure channels so that both parties can know it before encrypted messages can be sent over insecure channels. This is typically inconvenient, however. A better approach is, therefore, sought.

Public key cryptography overcomes the problem by eliminating the need for a single "secret" key. Here, each user of a public key cryptographic system has two mathematically-related keys, a "public key" and a secret or "private key." Operating in a complementary fashion, each key in the pair unlocks the code that the other key makes. Knowing the public key does not help deduce the corresponding private key, however. Accordingly, the public key can be published and widely disseminated across a communications network, such as the Internet, without in any way compromising the integrity of the private key. Anyone can use a recipient's public key to encrypt a message to that person, with the recipient, in turn, using his or her own corresponding private key to decrypt the message. One's private key, on the other hand, is kept secret, known only to the user.

Of particular interest to the present invention is use of cryptographic techniques for verifying or authenticating network communications, especially between software components. Cryptographic keys or digital "certificates" may be used to digitally sign a message or file and, in a complementary manner, to verify a digital signature. These "digital signatures" allow authentication of messages. When a user signs a message, a cryptographic program uses that user's own private key to create a digital signature that is unique to both the contents of the message and the user's private key. Any recipient can employ the user's public key to authenticate the signature. Since the signer, alone, possesses the private key that created that signature, authentication of a signature confirms that the message was actually sent by the signer, and that the message has not been subsequently altered by anyone else. Forgery of a signed message is not computationally feasible.

Authenticated communication is provided by a secured transport layer. In the presently-preferred embodiment, this secured transport layer is provided by PGP™ TLS (Transport Layer Security), which is available in run-time library form from PGPsdk™ (available from Pretty Good Privacy, Inc., a wholly-owned subsidiary of Network Associates, Inc.). PGPsdk is described in the documentation accompanying PGPsdk (including Inside the PGPsdk), the entire disclosure of which is hereby incorporated by reference. However, the Active Security Layer of the present invention may instead be implemented using other transport layers, such as SSL (Secure Socket Layer), TCP/IP (Transmission Control Protocol/Internet Protocol) with a security mechanism (e.g., third-party security mechanism), or the like. Cryptographic techniques and systems, including ones implementing public key cryptography, are well-documented in the technical, trade, and patent literature. For a general description, see e.g., Schneier, Bruce, *Applied Cryptography,* Second Edition, John Wiley & Sons, Inc., 1996. For a description focusing on the PGP™ implementation of public key cryptography, see e.g., Garfinkel, Simon, PGP™: *Pretty Good Privacy,* O'Reilly & Associates, Inc., 1995. The disclosures of each of the foregoing are hereby incorporated by reference.

During system operation, the Active Security Layer **350** (employing the PGPsdk™ run-time library) provides socket communication, establishment of communication sessions, exchanging of authentication certificates, and the like. At the time of secure communication, participating components may exchange respective digital certificates, each examining the certificate of the other for determining whether it has been signed by a party that is trusted. Here, the PKI has served as the provider of certificates. In the event that the certificates of respective components are trusted by the same party (i.e., PKI **360**, as configured by the system administrator), the components may now trust one another. Thus after successfully exchanging certificates (i.e., accepted as being trusted), communications between the components may proceed in a secure manner. Since the root of trust occurs at the PKI **360**, access to configuration of the PKI **360** itself for requesting/issuing certificates should be restricted (e.g., system administrator-level privileges).

With a basic understanding of the general application of cryptographic technique to communication, including the organization of the PGP™ key data structure, the reader has a foundation for understanding the teachings of the present invention for implementing an "active firewall" providing methodology for proactive management of network security.

C. General methodology

Referring now to FIGS. **5A–B**, the overall methodology of the present invention may be summarized by a flowchart **500**, which includes the following method steps. Step **501** indicates that, at the outset, the system is invoked with the context that the system has been configured by a system administrator for specifying which components of the system may participate in the process. Step **502** indicates an occurrence of an event of interest in the system, such as a sensor (e.g., scanner) detecting a compromise or vulnerability in the system. As a result of occurrence of the event, there now exists a need for one component (i.e., the sensor) to communicate in a secure, authenticated manner with another component (e.g., the arbiter or Event Orchestrator). Because of the configuration at step **501**, the Public Key Infrastructure of the system already stores in its repository signed digital certificates (i.e., signed by the system administrator) for the two components, so that the components can engage in a digital conversation (i.e., communication session).

Now, at step **503**, the sensor—acting as a "sender"—invokes the following substeps **511–517** for effecting authenticated communication with one or more listeners. As step **511**, the sender creates a "certogram"—that is, a packet of information describing the event which is organized into a format suitable for transmission. In the currently-preferred embodiment, a certogram may be constructed using attribute/value pairs in plain text, such as <attribute>=<value>, with a delimiter employed for separating one pair from the next. At step **512**, the sender determines which component(s) are its listeners. This determination is performed by simply enumerating those component(s) that have been specified in the system configuration to be listeners for this scanner. The components in the system may be specified by an IP (Internet Protocol) address or other identification scheme.

Step **513** illustrates the establishment of a socket connection. In the currently-preferred embodiment, this is performed through PGPTM TLS using a sequence of API (application programming interface) calls into the PGPsdk™ run-time library, as illustrated in Appendix A. Here, the component opens a socket connection (communication to a particular IP address on a particular port), binds that to a session, and then broadcasts a message to that port announcing its presence. At this point in the process, the communication socket is simply a conventional stream socket; communication is not yet authenticated. If a listener is present, the listener will respond with an acknowledgment accepting the socket connection. The receipt of an acknowledgment back from one or more listeners is indicated by step **514**. Now that the communication layer is established, the method may proceed to the next step, for exchanging certificates with the listener(s), as indicated by step **515**.

At step **516**, the respective sender/listener(s) components each validate the certificate received from the other. Validation may proceed in a conventional manner, for example using X.509 validation, including determining the level of trust and validity (including, for instance, the expiration, revocation, and disablement) of a given certificate. X.509 validation technique itself is documented in the technical literature. See, for example, the following RFC (Request for Comments) documents: RFC2459, RFC2585, RFC2559, RFC2552, RFC2538, RFC2528, RFC2527, RFC2511, RFC2510, and RFC2246, presently available on the Internet at http://www.faqs.org/rfcs/. The disclosures of the foregoing are hereby incorporated by reference.

If each respective component is able to successfully validate the certificate received from the other, secure communication ensues. From that point on, communication occurs in a secure, authenticated manner, with each message or blob being digitally signed or fingerprinted, for instance, using a cryptographic hash or message digest. This is indicated by step **517**. Any alteration to the message breaks the digital fingerprint and, therefore, may easily be detected. If desired, encryption may also be (optionally) applied to the communication messages. In those embodiments intended for export from the United States, however, encryption may be limited (e.g., as to key length) or removed.

Returning back to step **504**, the listener(s) decides whether to react to (e.g., act on, pass on, and/or otherwise process) the event reported by the certogram, or simply to ignore it. If the event is not to be acted on or otherwise processed, the method is done. Typically, however, the reported event maps to a script-defined event handler (e.g., illustrated in Appendix B below) in the listener (e.g., Event Orchestrator or EO) which, in turn, desires to notify yet another listener, the actor (e.g., firewall). Communication may therefore continue with the arbiter (EO) communicating with the target actor(s) in an authenticated manner in step **505**. As before, substeps **511–517** are invoked, this time with the arbiter (EO) as the sender and the actor (firewall) as the listener. The actor or firewall, upon receiving the certogram, may now (optionally) react to the event (i.e., undertake appropriate action), such as dynamically creating or modifying rules for appropriately handling the event, or it may choose to ignore the event (e.g., if the event is a duplicate of a previous event or if the event is covered by (or is a sub-set of) an existing firewall rule).

Although not shown as a separate step, in a preferred embodiment it is preferable to have the event first "filtered" through the arbiter (e.g., EO) as the sensor (e.g., scanner) will typically detect a multitude of events, many of which are not of interest to the actor (e.g., firewall). The extra level of filtering and correlation may be based on a company's security policy. Here, the company's security policy may be implemented as a script at the EO. Also, the extra level of filtering minimizes the reconfiguration of the firewall which, as it typically executes at ring 0 (i.e., highest privileged process), is often a computationally-expensive operation. In this manner, the firewall is not distracted from its main task of filtering communication packets. Note that this step replaces the present-day human step of reporting a network vulnerability to one's manager who, in turn, must then create a new firewall rule for addressing the vulnerability.

Appended herewith as Appendix A are annotated C/C++ source code listings providing further description of the present invention. A suitable development environment (compiler/linker) for compiling the source code is available from a variety of vendors, including Microsoft Visual C++ (available from Microsoft Corporation of Redmond, Wash.). Also appended herewith as Appendix B are annotated Visual Basic™ source code listings illustrating implementation of an exemplary handler routine for providing further description of the present invention. A suitable development environment (compiler/linker) for compiling the source code is Microsoft Visual Basic™ (available from Microsoft Corporation of Redmond, Wash.); however, the Visual Basic scripts for the Event Orchestrator may themselves be created with a text editor.

Further description of the network security system of the present invention is available in the documentation accompanying the commercial embodiment of the present invention, Network Associates'Gauntlet™ Active Firewall, which includes Gauntlet™ Firewall v5.0 (Windows NT and Solaris), CyberPatrol™, Telemate.Net QuickView Reporting, CyberCop™ Scanner v5.0, Event Orchestrator v1.0.2 (WinNT), Net Tools PKI Server v1.0 (Windows NT, DSA and RSA+DSA versions), Gauntlet™ VPN v5.0 (Windows NT and Solaris), and CyberCop™ Monitor v2.0 (when available), WebShield SMTP (Solaris v3.1.6, Windows NT v4.0.2). Gauntlet™ Active Firewall is available from Network Associates, Inc. of Santa Clara, Calif. (e.g., Part Nos./SKUs GAF-DRCT-NA-500-P, GAF-DRCT-NA-500-S1, and GAF-DRCT-NA-500-S), the disclosure of which is hereby incorporated by reference.

While the invention is described in some detail with specific reference to a single-preferred embodiment and certain alternatives, there is no intent to limit the invention to that particular embodiment or those specific alternatives. Thus, the true scope of the present invention is not limited to any one of the foregoing exemplary embodiments but is instead defined by the appended claims.

APPENDIX A

```
//
// Function:       CClient::PGPCreate( )
//
// Description:    This function creates the "sending" connection for Active
//                 Security. This function is used by the "sending" or "client"
//                 device (in Winsock terminology).
//
//                 In short this function does the work of calling to a server,
//                 opening a "client" or 'sending" connection AND do all the work
//                 proving that the two ends of this connection are properly
//                 Authenticated as per X.509 regulations.
//
// Procedure:
//                 1) we initialize the PGP SDK (creates memory, ensures enough
//                 available entropy, creates all TLS needed by socket.
//
//                 2) we initialize a PGP Socket connection with the
//                 Authentication-only cipher suite. We could add an encryption
//                 cipher at this point.
//
//                 3) we open up our local Key-Ring files and find the private key
//                 which matches the private key we installed AS with (stored as a
//                 byte string)
//
//                 4) we open the socket
//
//                 5) we establish a conection with the "server" or "listening"
//                 socket
//
//                 6) we create an in-memory "key-chain-set" with which we'll present
//                    (and exchange) our credentials with the remote (as yet) un-
//                    authenticated machine.
//
//                    This includes extracting the X.509 Signature Reference.
//
//                 7) we call ValidateClientKey( ) which:
//
//                    a) exchanges with the server our private key
//
//                    b) verifies that our private X.509 material has not been
//                       revoked by checking the CRL (Certificate Revocation List)
//
//                    c) verifies the authenticity of the remote key-pair
//
//                    d) verifies the PGP "validity" of the exchanged keys, this
//                       includes internal states such as PGP Validity, Expiration
//                       and so forth.
//
//                 8) if all is well (we have an authenticated connection) return 0
//
//

//
PGPError CClient::PGPCreate( )
{
     auto    PGPError                pgpErr
             = kPGPError_NoErr;
     auto    PGPUInt32               pgpui32InetAddr
       =     0;
     auto    PGPInt32                pgpi32SockRes
       =     0;
     auto    PGPSocketAddressInternet        pgpSockAddrIPSrv
     = {0};
     auto    PGPHostEntry*           hostEntry
             = NULL;
     auto    PGPSigRef               keySig
             = kInvalidPGPSigRef;
     auto    PGPKeySetRef            certChainKeySet
       =     kInvalidPGPKeySetRef;
     auto    PGPKeyRef               sigKey
             = kInvalidPGPKeyRef;
     // [Initialize the PGPsdk
                                     ]
     //
     m_pgpKSRefCli   = kInvalidPGPKeySetRef;
     m_pgpPrvKRefCli   = kInvalidPGPKeyRef;
     // [Do some PGP initialization
                                       ]
```

APPENDIX A-continued

```
    //
    pgpErr = PGPsdkInit( );
    if( IsPGPError( pgpErr ) )
        {
        Logger::Log(IDS_FAILPGPSDKINIT,EVENTLOG_ERROR_TYPE);
        goto __Done__PGPCreate;
        }
    pgpErr = PGPsdkNetworkLibInit( );
    if( IsPGPError( pgpErr ) )
        {
        Logger::Log(IDS_FAILPGPSDKNETLIBINIT,EVENTLOG_ERROR_TYPE);
        goto __Done__PGPCreate;
        }
    pgpErr = PGPsdkUILibInit( );
    if( IsPGPError( pgpErr ) )
        {
        Logger::Log(IDS_FAILPGPSDKUILIBINIT,EVENTLOG_ERROR_TYPE);
        goto __Done__PGPCreate;
        }
    pgpErr = PGPSocketsInit( );
    if( IsPGPError( pgpErr ) )
        {
        Logger::Log(IDS_FAILPGPSDKSOCKINIT,EVENTLOG_ERROR_TYPE);
        goto __Done__PGPCreate;
        }
    pgpErr = PGPSocketsCreateThreadStorage( &m_pgpSTSRefCli );
    if( IsPGPError( pgpErr ) )
        {
        Logger::Log(IDS_FAILPGPSDKTHREADSTOREINIT,EVENTLOG_ERROR_TYPE);
        goto __Done__PGPCreate;
        }
    // [Create a m_pgpConRefCli for all PGPsdk calls
                                    ]
    //
    pgpErr = NewContext( &m_pgpConRefCli );
    if( IsPGPError( pgpErr ) )
        {
        Logger::Log(IDS_FAILPGPSDKNEWCONTEXT,EVENTLOG_ERROR_TYPE);
        goto __Done__PGPCreate;
        }
    // [Open our key rings read-only
                                    ]
    //
    pgpErr =
PGPNewFileSpecFromFullPath(m_pgpConRefCli,m_szPubKRFNameCli,&m_pubKFSRefCli);
    if( IsPGPError( pgpErr ) )
        goto __Done__PGPCreate;
    pgpErr =
PGPNewFileSpecFromFullPath(m_pgpConRefCli,m_szPrivKRFNameCli,&m_prvKFSRefCli);
    if( IsPGPError( pgpErr ) )
        goto __Done__PGPCreate;
    pgpErr = PGPOpenKeyRingPair( m_pgpConRefCli
                                ,0
                                ,m_pubKFSRefCli
                                ,m_prvKFSRefCli
                                ,&m_pgpKSRefCli );
    if( IsPGPError( pgpErr ) )
        {
        Logger::Log(IDS_FAILPGPOPENKRPAIR,EVENTLOG_ERROR_TYPE);
        goto __Done__PGPCreate;
        }
    // [Locate our private key. We assume that it is the only private key in the
key]
    // [ring.
                                  ]
    //
    pgpErr = GetFirstPrivateKeyInSet( m_pgpKSRefCli, &m_pgpPrvKRefCli );
    if( IsPGPError( pgpErr ) )
        {
        Logger::Log(IDS_FAILPGPGET1STPRIVKEY,EVENTLOG_ERROR_TYPE);
        goto __Done__PGPCreate;
        }
    // [Create a TLS m_pgpConRefCli for all network calls. This needs to be done
    ]
    // [only once per app instance (that was the comment) ????
                   ]
    //
    pgpErr = PGPNewTLSContext( m_pgpConRefCli, &m_pgpTlsConRefCli );
    if( IsPGPError( pgpErr ) )
```

APPENDIX A-continued

```
    {
    Logger::Log(IDS_FAILPGPNEWTLSCONTEXT,EVENTLOG_ERROR_TYPE);
    goto __Done_PGPCreate;
    }
  // [Create a TLS session for this connection. This needs to be done
  once per ]
  // [connection.
                  ]
  //
  pgpErr = PGPNewTLSSession( m__pgpTlsConRefCli, &m__pgpTlsSessRefCli );
  if( IsPGPError( pgpErr ) )
    {
    Logger::Log(IDS__FAILPGPNEWTLSSESS,EVENTLOG__ERROR__TYPE);
    goto __Done__PGPCreate;
    }
  // [Setup options on the TLS session
                      ]
  //
  pgpErr = PGPtlsSetProtocolOptions( m__pgpTlsSessRefCli,kPGPtlsFlags__ClientSide );
  if( IsPGPError( pgpErr ) )
    {
    Logger::Log(IDS__FAILPGPSETPROTOCOLOPT,EVENTLOG__ERROR__TYPE);
    goto __Done__PGPCreate;
    }
  pgpErr = MakeCertChainKeySet (m__pgpConRefCli
                  ,m__pgpKSRefCli
                  ,&certChainKeySet
                  ,m__szRootCertKeyIdString
                  ,m__szProdCertKeyIdString
                  ,m__szPubKRFNameCli);
  if( IsPGPError( pgpErr ) )
    {
    Logger::Log(IDS__FAILPGPMAKECERTCHAINKEYSET,EVENTLOG__ERROR__TYPE);
    goto __Done__PGPCreate;
    }
  pgpErr = GetX509SigRef (certChainKeySet, &keySig,&sigKey);
  if( IsPGPError( pgpErr ) )
    {
    Logger::Log(IDS__FAILPGPGETX509SIGREF,EVENTLOG__ERROR__TYPE);
    goto __Done__PGPCreate;
    }
  pgpErr = PGPtlsSetLocalPrivateKey( m__pgpTlsSessRefCli
                      ,sigKey
                      ,keySig
                      ,certChainKeySet
                      ,PGPOPassphrase ( m__pgpConRefCli,
(LPCSTR)m__szPassPhrase )
                      ,PGPOLastOption( m__pgpConRefCli ) );
  if( IsPGPError( pgpErr ) )
    {
    Logger::Log(IDS__FAILPGPSETLOCALPRIVATEKEY,EVENTLOG__ERROR__TYPE);
    goto __Done__PGPCreate;
    }
  // [Set authentication suite only
                 ]
  //
  pgpErr = PGPtlsSetPreferredCipherSuite(
m__pgpTlsSessRefCli,kPGPtls__TLS__DHE__DSS__WITH__NULL__SHA );
  // [open the new socket and get a "template" socket reference . . .
              ]
  //
  m__pgpSocRefCli = PGPOpenSocket(
kPGPAddressFamilyInternet,kPGPSocketTypeStream,kPGPTCPProtocol );
  //
  if( !PGPSocketRefIsValid( m__pgpSocRefCli ) )
    {
    pgpErr = PGPGetLastSocketsError( );
    Logger::Log(IDS__FAILPGPOPENSOCKET,EVENTLOG__ERROR__TYPE);
    goto __Done__PGPCreate;
    }
  // [Lookup our server
                  ]
  //
  pgpui32InetAddr = PGPDottedToInternetAddress( m__szSrvrIPAddr );
  if( pgpui32InetAddr != kPGPSockets__Error )
    {
    hostEntry = PGPGetHostByAddress((char*)&pgpui32InetAddr
                      ,sizeof( PGPInternetAddress )
                      ,kPGPProtocolFamilyInternet );
```

APPENDIX A-continued

```
        }
    else
        {
        hostEntry = PGPGetHostByName( m__szSrvrIPAddr );
        }
    pgpSockAddrIPSrv.sin__family     = kPGPAddressFamilyInternet;
    pgpSockAddrIPSrv.sin__port       = PGPHostToNetShort( m__pgpi16SrvPort );
    // [If we were able 2 get the hostentry, use the IP address list from that. If
    ]
    // [not,use the IP address passed in by the caller.
                ]
    //
    if( hostEntry != NULL )
        {
        pgpSockAddrIPSrv.sin__addr = * ( (PGPInternetAddress*)
*hostEntry—>h__addr__list);
        }
    else
        {
        pgpSockAddrIPSrv.sin__addr = * ( (PGPInternetAddress*) &pgpui32InetAddr);
        }
    pgpi32SockRes = PGPConnect( m__pgpSocRefCli
                ,(PGPSocketAddress *) &pgpSockAddrIPSrv
                ,sizeof( pgpSockAddrIPSrv ) );
    if( pgpi32SockRes != kPGPSockets__Error )
        {
        // [Bind our TLS info to the m  pgpSocRefCli
                    ]
        //
        pgpErr = PGPSocketsEstablishTLSSession ( m__pgpSocRefCli, m__pgpTlsSessRefCli
);
        }
    else
        {
        Logger::Log(IDS__FAILPGPCONNECT,EVENTLOG__ERROR__TYPE);
        }
    if( IsntPGPError( pgpErr ) )
        {
        pgpErr = ValidateClientKey( m__pgpTlsSessRefCli );
        }
    else
        {
        Logger::Log(IDS__FAILPGPSOCKESTSESS,EVENTLOG__ERROR__TYPE);
        }
__Done__PGPCreate:
    if( certChainKeySet != NULL )
        PGPFreeKeySet ( certChainKeySet );
    return pgpErr;
}
//

// Function: CClient::ValidateClientKey( )
//
// Description:
//      Get the client's key and validate it. There are three ways to validate the
//      key. First, we can lookup the key in a list of known keys maintained by
//      the server. Second, we can directly validate the key is signed by a known
//      key. Third, we can trust a known CA key and check the validity of the
//      client key. If it is valid, it has been signed by the CA key or another
//      trusted key. We're employing method three here.
//
// Procedure:
//
//      1) Get the remote machine's client key.
//
//      2) Check that the key has not expired or is in some way invalid/disabled
//
//      3) do the PGP work of determining if all the things that will make this KR
//         pair acceptable (enough for an authenticated connection)
//
//      4) If the validity of the key does not meet ALL our criteria then we'll log
//         this as an unacceptable connection.
//
//      5) or else return all is well.
//
//

//
PGPError CClient::ValidateClientKey(PGPtlsSessionRef tlsSR)
```

APPENDIX A-continued

```
{
    auto   PGPError        pgpErr        = kPGPError_NoErr;
    auto   PGPKeyRef       clientKey     = kInvalidPGPKeyRef;
    auto   PGPKeySetRef    clientSetKey  = kInvalidPGPKeySetRef;
    auto   PGPBoolean         isExpired;
    auto   PGPBoolean         isRevoked;
    auto   PGPBoolean         isDisabled;
    pgpErr = PGPtlsGetRemoteAuthenticatedKey( tlsSR, &clientKey,&clientSetKey );
    //
    if( IsntPGPError( pgpErr ) )
      {
      // Validate the key is not expired, revoked, etc
      //
      pgpErr = PGPGetKeyBoolean( clientKey, kPGPKeyPropIsRevoked, &isRevoked );
      if( IsntPGPError( pgpErr ) )
          {
          pgpErr = PGPGetKeyBoolean( clientKey
                                  ,kPGPKeyPropIsExpired
                                  ,&isExpired );
          if( IsntPGPError( pgpErr ) )
            {
            pgpErr = PGPGetKeyBoolean( clientKey
                                    ,kPGPKeyPropIsDisabled
                                    ,&isDisabled);
            }
          }
      if( IsntPGPError( pgpErr ) )
          {
          if( isExpired || isRevoked || isDisabled )
            {
            Logger::Log(IDS_CERTISNOTVALID,EVENTLOG_ERROR_TYPE);
            pgpErr = kPGPError_ServerAuthorizationFailed;
            }
          }
      if( IsntPGPError( pgpErr ) )
          {
          auto    PGPKeySetRef    combinedSet
    = kInvalidPGPKeySetRef;
          // Create a combined key set for computing validity
          //
          pgpErr = PGPNewKeySet( m_pgpConRefCli, &combinedSet );
          if( IsntPGPError( pgpErr ) )
            {
            auto    PGPKeySetRef    clientKeySet
    = kInvalidPGPKeySetRef;
            pgpErr = PGPNewSingletonKeySet( clientKey, &clientKeySet );
            if( IsntPGPError( pgpErr ) )
              {
              pgpErr = PGPAddKeys( m_pgpKSRefCli, combinedSet );
              if( IsntPGPError( pgpErr ) )
                {
                pgpErr = PGPAddKeys( clientKeySet, combinedSet );
                if( IsntPGPError( pgpErr ) )
                  {
                  pgpErr = PGPCheckKeyRingSigs( combinedSet,
                    combinedSet, FALSE, NULL, 0);
                  if( IsntPGPError( pgpErr ) )
                    {
                    pgpErr = PGPPropagateTrust( combinedSet );
                    }
                  }
                }
              // We now have a set containing the remote key and all trust and
validity
              // computation has been done. Get the validity of the remote key
and error
              // if it is not completely valid
              //
              if( IsntPGPError( pgpErr ) )
                {
                auto    PGPInt32    validity;
                pgpErr = PGPGetKeyNumber ( clientKey, kPGPKeyPropValidity,
&validity );
                if( IsntPGPError( pgpErr ) )
                  {
                  if( (PGPValidity) validity != kPGPValidity_Complete )
                    {
                    Logger::Log(IDS_CERTISNOTVALID,EVENTLOG_ERROR_TYPE);
                    pgpErr = kPGPError_ServerAuthorizationFailed;
```

APPENDIX A-continued

```
                        }
                    }
                }
              PGPFreeKeySet( clientKeySet );
              }
            PGPFreeKeySet ( combinedSet );
            }
          }
//      PGPFreeKeySet( clientSetKey );
}
    if( IsntPGPError( pgpErr ) )
      Logger::Log(IDS_CERTISVALID,EVENTLOG_INFORMATION_TYPE);
    return( pgpErr );
}
//  ______________________________________________________________________

// Function: PGPTimedSockets::TimedAccept( )
// Description:
//      This function is used by the server or "listening" Active Security device.
//      This mechanism allows multiple simultaneous attempts at connection to the
//      listener.
//
//      This function is part of a multi-threaded procedure launched when the
//      Listener is created.
//
// Procedure:
//
//      1) Depending on whether we're instructed to wait forever or not we setup
//         waiting-state structures that dictate how we will accept sockets.
//
//      2) We select any found open waiting socket
//
//      3) We accept same
//
//
// Remarks:
//      This function merely "accepts" a "standard" socket connection. This is not
//      where we determine the authenticity/acceptability of the connection. This
//      function will accept ANY Winsock compliant requests to connect to this
//      Listener.
//
//      It is not until later when we will do all the cryptography to determine if
//      the remote machine has the proper X.509 and Active Security credentials to
//      keep connection alive.
//
//      NO ACTIVE SECURITY DATA CAN BE TRANSFERED ON THIS CONNECTION JUST BECAUSE WE
//      ACCEPT IT. THIS CONNECTION STILL MUST BE AUTHENTICATED BEFORE THIS "STREAM"
//      IS MADE AVAILABLE TO THE SECURECOMM API'S.
//
//  ______________________________________________________________________

//
PGPSocketRef PGPTimedSockets::TimedAccept(PGPSocketRef socket, long timeout,
PGPError &err)
{
    PGPSocketSet readSet;
    PGPSocketRef currSock;
    PGPSocketsTimeValue timeOut;
    PGPSOCKETSET_ZERO(&readSet);
    PGPSOCKETSET_SET (socket, &readSet);
    if (timeout == INFINITE)
    err = PGPSelect(FD_SETSIZE,&readSet,NULL,NULL,NULL);
    else
    {
      timeOut.tv_sec = (long) (timeout/1000);
      timeOut.tv_usec = (long)0;
      err = PGPSelect(FD_SETSIZE,&readSet,NULL,NULL,&timeOut);
    }
    if (!err)
    {
      err = SOCKET_TIME_OUT;
      return 0;
    }
    if (err == 1)
    {
      currSock = PGPAccept(socket,NULL,NULL);
      if(currSock == NULL || (PGPInt32) currSock == kPGPSockets_Error)
        err = INVALID_SOCKET;
      else
```

APPENDIX A-continued

```
        err = 0;
    }
    return currSock;
}
```

APPENDIX B

```
' File:                CertogramMHD.vbs
' Version:             1.0
' Last Modified:       April 5, 1999
' Written By:          Network Associates, Inc.
' Copyright:           1999 Network Associates, Inc. All rights reserved.
'
' Description:
' This script will populate a McAfee HelpDesk (MHD) trouble ticket with information
' about a vulnerability detected by CyberCop Scanner. This is performed by
' extracting information from variables associated with an incoming Certogram and
' creating a customizable string variable, NOTE. This variable is used by the MHD
' Action Server in Event Orchestrator to populate the “Action Note” of a trouble
' ticket.
' Note:   Network Associates recommends that you use these scripts during the
'         “Generate” stage vs. the “Process” stage within Event Orchestrator for
'         optimal perfomance.
' Extract the product name and version that found the security issue
strProductID = EventObject.GetData (“CG_PRODUCT_VERSION”)
' Extract the vulnerability ID, severity of the problem and what action is
' recommended by the product that detected the vulnerability
strVulCode = EventObject.GetData (“CG_VULNERABILITY_ID”)
strSeverity = EventObject.GetData (“CG_SEVERITY”)
strAction = EventObject.GetData (“CG_ACTION”)
' Extract the IP address, port number and protocol ID associated with the system
' which has the vulnerability
strMachine = EventObject.GetData (“CG_MACHINE”)
strPort = EventObject.GetData (“CG_PORT”)
strProtocol = EventObject.GetData (“CG_PROTOCOL_ID”)
' Extract information about the vulnerability, including suggestions on how to fix
it
strShortDescription = EventObject.GetData (“CG_SHORT_DESCRIPTION”)
strLongDescription = EventObject.GetData (“CG_VULNERABILITY_DESCRIPTION”)
strConcerns = EventObject.GetData (“CG_SECURITY_CONCERNS”)
strSuggestions = EventObject.GetData (“CG_SUGGESTIONS”)
strOtherSources = EventObject.GetData (“CG_OTHER_INFORMATION_SOURCES”)
' Build string to be placed in the “Action Note” field of MHD trouble ticket
CRLF = Chr(13) + Chr(10)
strNote = “** ATTENTION **” + CRLF
strNote = strNote + “** ” + strProductID + “ detected a security vulnerability (” +
strVulCode + “) in the network. **” + CRLF + CRLF
strNote = strNote + “_____________________” + CRLF
strNote = strNote + “HOST INFORMATION” + CRLF
strNote = strNote + “IP Address: ” + strMachine + CRLF
strNote = strNote + “Port Number: ” + strPort + CRLF + CRLF
strNote = strNote + “__________________________” + CRLF
strNote = strNote + “INCIDENT INFORMATION” + CRLF
strNote = strNote + “The severity was: ”
' Determine severity of event and include it in the trouble ticket
if (strSeverity = “2”) then
  strNote = StrNote + “HIGH” + CRLF
elseif (strSeverity = “1”) then
  strNote = StrNote + “MEDIUM” + CRLF
elseif (strSeverity = “0”) then
  strNote = StrNote + “LOW” + CRLF
end if
strNote = strNote + “The action code was: BLOCK PORT” + CRLF
strNote = strNote + “The protocol used was: ”
' Determine which protocol the vulnerability is associated with and include it in
the trouble ticket
if (strProtocol = “0”) then
  strNote = StrNote + “IP” + CRLF
elseif (strProtocol = “1”) then
  strNote = StrNote + “ICMP” + CRLF
elseif (strProtocol = “2”) then
  strNote = StrNote + “IGMP” + CRLF
elseif (strProtocol = “3”) then
  strNote = StrNote + “GGP” + CRLF
elseif (strProtocol = “6”) then
  strNote = StrNote + “TCP” + CRLF
```

APPENDIX B-continued

```
elseif (strProtocol = “12”) then
  strNote = StrNote + “PUP” + CRLF
elseif (strProtocol = “17”) then
  strNote = StrNote + “UDP” + CRLF
end if
' Insert information about the vulnerability into the trouble ticket
strNote = strNote + “______________________________” + CRLF
strNote = strNote + “VULNERABILITY INFORMATION” + CRLF
strNote = strNote + “Short Description:” + CRLF
strNote = strNote + strShortDescription + CRLF + CRLF
strNote = strNote + “Vulnerability Description:” + CRLF
strNote = strNote + strLongDescription + CRLF + CRLF
strNote = strNote + “Security Concerns:” + CRLF
strNote = strNote + strConcerns + CRLF + CRLF
strNote = strNote + “Suggestions:” + CRLF
strNote = strNote + strSuggestions + CRLF + CRLF
strNote = strNote + “Other Information Sources:” + CRLF
strNote = strNote + strOtherSources + CRLF
' Set the “NOTE” variable so the MHD Action Server will populate the trouble ticket
EventObject.AddData “NOTE”, strNote
```

What is claimed is:

1. In a computer network system comprising a plurality of software components, a method for providing network security using authenticated communication between software components of the system, the method comprising:

specifying first, second, and third software components that may participate in authenticated communication, including creating a digital certificate for each software component;

detecting by the first component a security-related event of interest that occurs in the system;

initiating authenticated communication between the first software component and the second software component, so that the first software component may report the event to the second software component;

initiating authenticated communication between the second software component and the third software component, so that the second software component may indicate to the third software component how to handle the event; and

handling the event at the third software component in the manner indicated by the second software component, so that the event is automatically handled by the system.

2. The method of claim **1**, wherein each of the software components operates on a separate computer connected to the computer network system.

3. The method of claim **1**, wherein said specifying step includes:

receiving input from a user having system administrator privileges specifying which software components may participate in authenticated communication.

4. The method of claim **3**, wherein said input includes:

digitally signing a digital certificate of each software component permitted to participate in authenticated communication with a digital certificate for the user having system administrator privileges.

5. The method of claim **1**, wherein authenticated communication is initiated between components by:

exchanging digital certificates of the respective software components, and

if the digital certificate of each respective software component has been signed by an entity that the other software component trusts, establishing authenticated communication between the two software components.

6. The method of claim **5**, wherein each digital certificate created is stored in a central repository.

7. The method of claim **1**, wherein authenticated communication provided by the method includes:

associating a digital fingerprint with each message that occurs during authenticated communication.

8. The method of claim **7**, wherein said digital fingerprint includes a cryptographic hash.

9. The method of claim **7**, wherein said digital fingerprint includes a message digest.

10. The method of claim **7**, wherein the digital fingerprint for a given message is based, at least in part, on a digital certificate for the respective component that created the given message.

11. The method of claim **7**, wherein authenticated communication provided by the method includes:

authenticating a given message of the authenticated communication using the digital fingerprint specifically computed for the given message.

12. The method of claim **1**, further comprising:

encrypting communication between the software components.

13. The method of claim **12**, wherein said step of encrypting communication includes:

encrypting messages from one component to a digital certificate for the other component.

14. The method of claim **1**, wherein information about an event occurring in the system is transmitted as a certogram.

15. The method of claim **14**, wherein each certogram comprises information organized into attribute/value format.

16. The method of claim **1**, wherein authenticated communication provided by the method includes:

validating a digital certificate for each component participating in authenticated communication.

17. The method of claim **16**, wherein said validating step includes determining selected ones of expiration, revocation, and disablement for each digital certificate being validated.

18. The method of claim **1**, wherein the second software component includes an event handler for instructing the third software component how to appropriately handle the event.

19. The method of claim **1**, wherein said event occurring in the system comprises a detected vulnerability.

20. The method of claim **19**, wherein said system includes a firewall as said third component and wherein said event handler instructs the firewall to create a new firewall rule for appropriately handling the detected vulnerability.

21. A method for providing automated network security for a network system, the method comprising:

providing a configurable firewall capable of limiting access to the network system, a sensor for detecting vulnerabilities in the network system, and an arbiter for specifying reconfiguration of the firewall for handling vulnerabilities detected by the sensor;

specifying that the firewall, the sensor, and the arbiter may participate in authenticated communication;

detecting by the sensor a particular vulnerability in the network system;

establishing an authenticated communication session between the sensor and the arbiter for transmitting information about the particular vulnerability from the sensor to the arbiter; and

establishing an authenticated communication session between the arbiter and the firewall for transmitting instructions for handling the particular vulnerability from the arbiter to the firewall, such that the particular vulnerability may be handled in an automated manner.

22. The method of claim 21, wherein the firewall, the sensor, and the arbiter each operates on a separate computer connected to the network system.

23. The method of claim 21, wherein said specifying step includes:

receiving input from a user having system administrator privileges specifying which particular components in the system may participate in authenticated communication.

24. The method of claim 23, wherein said specifying step further includes:

creating a digital certificate for the user having system administrator privileges and creating a digital certificate for each component that is permitted to participate in authenticated communication;

digitally signing the digital certificate of each component permitted to participate in authenticated communication with the digital certificate for the user having system administrator privileges.

25. The method of claim 24, wherein each digital certificate comprises a PGP-compatible key.

26. The method of claim 23, wherein each digital certificate created is stored in a central repository.

27. The method of claim 21, wherein authenticated communication is established between components by authenticating messages communicated between those components using digital fingerprints.

28. The method of claim 27, wherein each digital fingerprint comprises a cryptographic hash.

29. The method of claim 27, wherein each digital fingerprint comprises a message digest.

30. The method of claim 27, wherein the digital fingerprint for a given message is based, at least in part, on a digital certificate of the respective component that created the given message.

31. The method of claim 27, wherein authenticated communication is established between components by authenticating a given message using a digital fingerprint specifically computed for the given message.

32. The method of claim 21, further comprising:

encrypting communication between components of the network system.

33. The method of claim 32, wherein said step of encrypting communication includes:

encrypting messages from one component to a digital certificate created for the other component.

34. The method of claim 21, wherein said information about the particular vulnerability comprises information organized into attribute/value format.

35. The method of claim 21, wherein authenticated communication is established between two components by exchanging digital certificates of each component with the other and thereafter establishing trust and validity for each digital certificate so exchanged.

36. The method of claim 35, wherein establishing validity includes determining selected ones of expiration, revocation, and disablement for each digital certificate being validated.

37. The method of claim 21, wherein the particular vulnerability comprises detection of unauthorized access to the network system.

38. The method of claim 21, wherein the particular vulnerability comprises detection of an unauthorized mail server on the network system.

39. The method of claim 21, wherein the particular vulnerability comprises detection of an unauthorized FTP (File Transport Protocol) server on the network system.

40. The method of claim 21, wherein the particular vulnerability comprises detection of an unauthorized writeable directory on the network system.

41. A system providing automatically-reconfigurable security for a computer network, the system comprising:

a configurable firewall component providing security to the computer network;

a sensor component for detecting security-related events that occur in the computer network;

an arbiter component for specifying reconfiguration of the firewall component for handling at least some of the security-related events detected by the sensor component; and

a communication layer, configured to specify that the firewall component, the sensor component, and the arbiter component may participate in authenticated communication, so that the firewall component may be automatically reconfigured by the arbiter component to handle a particular security-related event that has been detected by the sensor component.

42. The system of claim 41, wherein the firewall component, the sensor component, and the arbiter component each operates on a separate computer connected to the computer network.

43. The system of claim 41, wherein said communication layer may be configured from input from a user having system administrator privileges that specify which particular components in the system may participate in authenticated communication.

44. The system of claim 43, wherein said system operates, in response to said input, to create a digital certificate for the user having system administrator privileges, to create a digital certificate for each component that is permitted to participate in authenticated communication, and to digitally sign the digital certificate of each component permitted to participate in authenticated communication with the digital certificate for the user having system administrator privileges.

45. The system of claim 44, wherein each digital certificate comprises a PGP-compatible key.

46. The system of claim 44, wherein each digital certificate created is stored in a central repository.

47. The system of claim 41, wherein authenticated communication is established between components by authen-

ticating messages communicated between those components using digital fingerprints.

48. The system of claim **47**, wherein each digital fingerprint comprises a cryptographic hash.

49. The system of claim **47**, wherein each digital fingerprint comprises a message digest.

50. The system of claim **47**, wherein the digital fingerprint for a given message is based, at least in part, on a digital certificate of the respective component that created the given message.

51. The system of claim **47**, wherein authenticated communication is established between components by authenticating a given message using a digital fingerprint specifically computed for the given message.

52. The system of claim **41**, wherein said communication layer provides encryption of communication between components of the system.

53. The system of claim **52**, wherein encrypted messages sent to a particular component are encrypted to the public key certificate of that component.

54. The system of claim **41**, wherein information about security-related events is communicated in attribute/value format.

55. The system of claim **41**, wherein authenticated communication is established between two components by exchanging digital certificates of each component with the other and thereafter establishing trust and validity for each digital certificate so exchanged.

56. The system of claim **55**, wherein establishing validity includes determining selected ones of expiration, revocation, and disablement for each digital certificate being validated.

57. The system of claim **41**, wherein the particular security-related event comprises detection, of unauthorized access to the computer network.

58. The system of claim **41**, wherein the particular security-related event comprises detection of an unauthorized mail server on the computer network.

59. The system of claim **41**, wherein the particular security-related event comprises detection of an unauthorized FTP (File Transport Protocol) server on the computer network.

60. The system of claim **41**, wherein the particular security-related event comprises detection of an unauthorized writeable directory on the computer network.

* * * * *

US005950195A

United States Patent [19]

Stockwell et al.

[11] **Patent Number: 5,950,195**

[45] **Date of Patent: *Sep. 7, 1999**

[54] **GENERALIZED SECURITY POLICY MANAGEMENT SYSTEM AND METHOD**

[75] Inventors: **Edward B. Stockwell**, St. Paul; **Alan E. Klietz**, Fridley, both of Minn.

[73] Assignee: **Secure Computing Corporation**, Roseville, Minn.

[*] Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

[21] Appl. No.: **08/715,668**

[22] Filed: **Sep. 18, 1996**

[51] **Int. Cl.**^{6} **G06F 17/30**

[52] **U.S. Cl.** **707/4**; 707/3; 707/100; 707/102; 707/103; 707/104; 707/513

[58] **Field of Search** 380/25, 23; 395/615, 395/187.01, 206.12; 707/3, 4, 100, 102, 103, 104, 513

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,956,615	5/1976	Anderson et al.	380/49
4,104,721	8/1978	Markstein et al.	364/200
4,177,510	12/1979	Appell et al.	364/200
4,442,484	4/1984	Childs, Jr. et al.	364/200
4,584,639	4/1986	Hardy	364/200
4,621,321	11/1986	Boebert et al.	364/200
4,648,031	3/1987	Jenner et al.	364/200
4,701,840	10/1987	Boebert et al.	364/200
4,713,753	12/1987	Boebert et al.	380/4
4,870,571	9/1989	Frink	364/200
4,885,789	12/1989	Burger et al.	380/25
4,888,801	12/1989	Foster et al.	380/21
4,914,568	4/1990	Kodosky et al.	364/200
5,093,914	3/1992	Coplien et al.	395/700
5,124,984	6/1992	Engel	370/94.1

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0 554 182 A1	4/1993	European Pat. Off.	H04L 29/06
0 743 777 A2	11/1996	European Pat. Off.	H04L 29/06
96/13113	5/1996	WIPO	H04L 29/06
96/31035	10/1996	WIPO	H04L 12/24
97/13340	4/1997	WIPO	H04L 9/00
97/16911	5/1997	WIPO	H04L 29/06
97/23972	7/1997	WIPO	H04L 9/00
97/26731	7/1997	WIPO	H04L 9/00
97/26734	7/1997	WIPO	H04L 9/00
97/26735	7/1997	WIPO	H04L 9/00
97/29413	8/1997	WIPO .	

OTHER PUBLICATIONS

Hong, Toue, and Leifer, "Personal Electronic Notebook with Sharing", IEEE/IEE Publications, pp. 88–94, Apr. 20, 1995.

Steffen Stempel, "IpAcess—An Internet Service Access System for Firewall Installations", IEEE, pp. 31–41, 1995.

Greenwald, Singhal, Stone, and Cheriton, "Designing an Academic Firewall: Policy, Practice, and Experience With Surf", pp. 79–92, 1996.

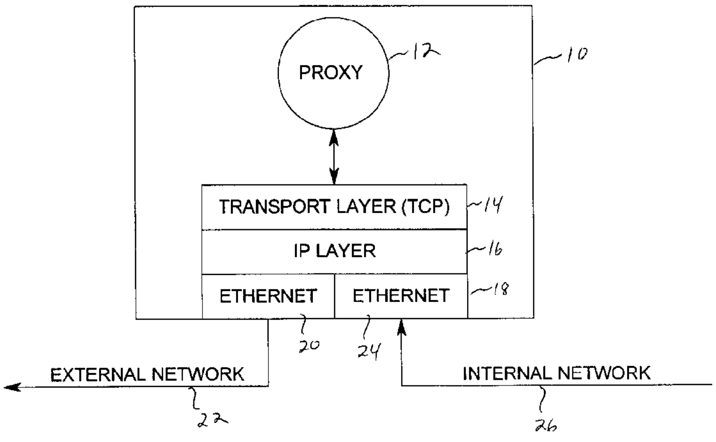
(List continued on next page.)

Primary Examiner—Thomas G. Black
Assistant Examiner—Cheryl Lewis
Attorney, Agent, or Firm—Schwegman, Lundberg, Woessner & Kluth, P.A.

[57] **ABSTRACT**

A system and method for regulating the flow of internetwork connections through a firewall having a network protocol stack which includes an Internet Protocol (IP) layer. A determination is made of the parameters characteristic of a connection request, including a netelement parameter characteristic of where the connection request came from. A query is generated and a determination is made whether there is a rule corresponding to that query. If there is a rule corresponding to the query, a determination is made whether authentication is required by the rule. If authentication is required by the rule, an authentication protocol is activated and the connection is activated if the authentication protocol is completed successfully.

19 Claims, 5 Drawing Sheets



U.S. PATENT DOCUMENTS

5,153,918	10/1992	Tuai	380/25
5,204,961	4/1993	Barlow	395/725
5,228,083	7/1993	Lozowick et al.	380/9
5,263,147	11/1993	Francisco et al.	395/425
5,272,754	12/1993	Boebert	380/25
5,276,735	1/1994	Boebert et al.	380/21
5,303,303	4/1994	White	380/49
5,305,385	4/1994	Schanning et al.	380/49
5,311,593	5/1994	Carmi	380/23
5,329,623	7/1994	Smith et al.	395/275
5,333,266	7/1994	Boaz et al.	395/200
5,355,474	10/1994	Thuraisngham et al.	395/600
5,414,833	5/1995	Hershey et al.	395/575
5,416,842	5/1995	Aziz	380/30
5,485,460	1/1996	Schrier et al.	370/94.1
5,511,122	4/1996	Atkinson	380/25
5,530,758	6/1996	Marino, Jr. et al.	380/49
5,548,646	8/1996	Aziz et al.	380/23
5,550,984	8/1996	Gelb	395/200.17
5,566,170	10/1996	Bakke et al.	370/60
5,583,940	12/1996	Vidrascu et al.	380/49
5,586,260	12/1996	Hu	395/200.2
5,604,490	2/1997	Blakley, III et al.	340/825.31
5,606,668	2/1997	Shwed	395/200.11
5,615,340	3/1997	Dai et al.	395/200.17
5,619,648	4/1997	Canale et al.	395/200.01
5,623,601	4/1997	Vu	395/187.01
5,636,371	6/1997	Yu	395/500
5,644,571	7/1997	Seaman	370/401
5,671,279	9/1997	Elgamal	380/23
5,673,322	9/1997	Pepe et al.	380/49
5,684,951	11/1997	Goldman et al.	395/188.01
5,689,566	11/1997	Nguyen	380/25
5,699,513	12/1997	Feigen et al.	395/187.01
5,706,507	1/1998	Schloss	395/615
5,708,780	1/1998	Levergood et al.	395/200.12
5,720,035	2/1998	Allegre et al.	395/200.06
5,724,425	3/1998	Chang et al.	380/25
5,781,550	7/1998	Templin et al.	370/401

OTHER PUBLICATIONS

Bill Gassman, "Internet Security, and Firewalls Protection on the Internet", IEEE, pp. 93–107, 1996.

S. Cobb, "Establishing fiewall policy", IEEE, pp. 198–205, 1996.

Steven M. Bellovin and William R. Cheswick, "Network Firewalls", IEEE, pp. 50–57, Sep. 1994.

Lee J. White and Hareton K.N. Leung, "A Firewall Concept for both Control–Flow and Data–Flow in Regression Integration Testing", IEEE, pp. 262–271, 1992.

"100% of Hackers Failed to Break Into One Internet Site Protected by Sidewinder", News Release, Secure Computing Corporation, (Feb. 16, 1995).

"Internet Security System Given 'Product of the Year' Award", News Release, Secure Computing Corporation (Mar. 28, 1995).

"Satan No Threat to Sidewinder™", News Release, Secure Computing Corporation (Apr. 26, 1995).

"Answers to Frequently Asked Questions About Network Security", Secure Computing Corporation, 41 p. (1994).

Adam, J.A., "Meta–matrices", *IEEE Spectrum,* 26–27 (Oct. 1992).

Adam, J.A., "Playing on the Net", *IEEE Spectrum,* 29 (Oct. 1992).

Ancilotti, P., et al., "Language Features for Access Control", *IEEE Transactions on Software Engineering,* SE–9, 16–25 (Jan. 1983).

Badger, L., et al., "Practical Domain and Type Enforcement for UNIX", *Proceedings of the 1995 IEEE Symposium on Security and Privacy,* Oakland, CA, 66–77 (May 8–10, 1995).

Belkin, N.J., et al., "Information Filtering and Information Retrieval: Two Sides of the Same Coin?", *Communications of the ACM,* 35, 29–28 (Dec. 1992).

Bellovin, S.M., et al., "Network Firewalls", *IEEE Communications Magazine,* 32, 50–57 (Sep. 1994).

Bevier, W.R., et al., "Connection Policies and Controlled Interference", *Proceedings of the 8th IEEE Computer Security Foundations Workshop,* Kenmare, County Kelly, Ireland, 167–176 (Jun. 13–15, 1995).

Bowen, T.F., et al., "The Datacycle Architecture", *Communications of the ACM,* 35, 71–81 (Dec. 1992).

Bryan, J., "Firewalls For Sale", *BYTE,* pp. 99–100, 102 and 104 (Apr. 1995).

Damashek, M., "Gauging Similarity with n–Grams: Language–Independent Categorization of Text", *Science,* 267, 843–848 (Feb. 10, 1995).

Dillaway, B.B., et al., "A Practical Design for A Multilevel Secure Database Management System", *American Institute of Aeronautics and Astronautics, Inc.,* pp. 44–57 (Dec. 1986).

Fine, T., et al., "Assuring Distributed Trusted Mach", *Proceedings of the 1993 IEEE Computer Society Symposium on Research in Security and Privacy,* 206–218 (1993).

Foltz, P.W., et al., "Personalized Information Delivery: An Analysis of Information Filtering Methods", *Communications of the ACM,* 35, 51–60 (Dec. 1992).

Goldberg, D., et al., "Using Collaborative Filtering to Weave an Information Tapestry", *Communications of the ACM,* 35, 61–70 (Dec. 1992).

Grampp, F.T., "UNIX Operating System Security", *AT&T Bell Laboratories Technical Journal,* 63, 1649–1672 (Oct. 1984).

Haigh, J.T., et al., "Extending the Non–Interference Version of MLS for SAT", *Proceedings of the 1986 IEEE Symposium on Security and Privacy,* Oakland, CA, 232–239 (Apr. 7–9, 1986).

Kent, S.T., "Internet Privacy Enhanced Mail", *Communications of the ACM,* 36, 48–60 (Apr. 1993).

Lampson, B.W., "Dynamic Protection Structures", *AFIPS Conference Proceedings,* vol. 35, 1969 Fall Joint Computer Conference, Las Vegas, NV, 27–38 (Nov. 18–20, 1969).

Lee, K.–C., et al., "A Framework for Controlling Cooperative Agents", *Computer,* 8–16 (Jul. 1993).

Loeb, S., "Architecting Personalized Delivery of Multimedia Information", *Communications of the ACM,* 35, 39–50 (Dec. 1992).

Loeb, S., et al., "Information Filtering," *Communications of the ACM,* 35, 26–28 (Dec. 1992).

Merenbloom, P., "Network 'Fire Walls' Safeguard LAN Data from Outside Intrusion", *InfoWorld,* p. 69 (Jul. 25, 1994).

Obraczka, K., et al., "Internet Resource Discovery Services", *Computer,* 26, 8–22 (Sep. 1993).

Press, L., "The Net: Progress and Opportunity", *Communications of the ACM,* 35, 21–25 (Dec. 1992).

Schroeder, M.D., et al., "A Hardware Architecture for Implementing Protection Rings", *Communications of the ACM,* 15, 157–170 (Mar. 1972).

Schwartz, M.F., "Internet Resource Discovery at the University of Colorado", *Computer,* 26, 25–35 (Sep. 1993).

Smith, R.E., "Sidewinder: Defense in Depth Using Type Enforcement", *International Journal of Network Management*, 219–229, (Jul.–Aug. 1995).

Thomsen, D., "Type Enforcement: The New Security Model", *Proceedings of the SPIE, Multimedia: Full–Service Impact on Business, Education and the Home*, vol. 2617, Philadelphia, PA, 143–150 (Oct. 23–24, 1995).

Warrier, U.S., et al., "A Platform for Heterogeneous Interconnection Network Management", *IEEE Journal on Selected Areas in Communications*, 8, 119–126 (Jan. 1990).

Wolfe, A, "Honeywell Builds Hardware for Computer Security", *Electronics*, 14–15 (Sep. 2, 1985).

International Search Report , PCT Application No. PCT/US 95/12681, 8 p. (mailed Apr. 9, 1996).

"Sidewinder Internals", Product information, Securing Computing Corporation, 16 p. (Oct. 1994).

"Special Report: Secure Computing Corporation and Network Security", *Computer Select*, 13 p. (Dec. 1995).

Atkinson, R., "IP Authentication Header", Network Working Group, Request For Comment No. 1826, http//ds.internic.net/rfc/rfc1826.txt, 9 p. (Aug. 1995).

Atkinson, R., "IP Encapsulating Security Payload (ESP)", Network Working Group, Request For Comment No. 1827, http//ds.internic.net/rfc/rfc1827.txt, 12 p. (Aug. 1995).

Atkinson, R., "Security Architecture for the Internet Protocol", Network Working Group, Reqest for Comment No. 1825, http//ds.internic.net/rfc/rfc1825.txt, 21 p. (Aug. 1995).

Baclace, P.E., "Competitive Agents for Information Filtering", *Communications of the ACM*, 35, 50 (Dec. 1992).

Karn, P., et al., "The ESP DES–CBC Transform", Network Working Group, Request for Comment No. 1829, http//ds.internic.net/rfc/rfc1829.txt, 9 p. (Aug. 1995).

McCarthy, S.P., "Hey Hackers! Secure Computing Says You Can't Break into This Telnet Site", *Computer Select*, 2 p. (Dec. 1995).

Metzger, P., et al., "IP Authentication using Keyed MD5", Network Working Group, Request for Comments No. 1828, http//ds.internic.net/rfc/rfc1828.txt, 5 p. (Aug. 1995).

Peterson, L.L., et al., In: *Computer Networks*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, pp. 218–221, 284–286 (1996).

Smith, R.E., "Constructing a High Assurance Mail Guard", *Proceedings of the 17th Annual National Computer Security Conference*, pp. 247–253)Oct. 1994).

Stadnyk, I., et al., "Modeling User's Interests in Information Filters", *Communications of the ACM*, 35, 49–50 (Dec. 1992).

Stevens, C., "Automating the Creation of Information Filters", *Communications of the AMC*, 35, 48 (Dec. 1992).

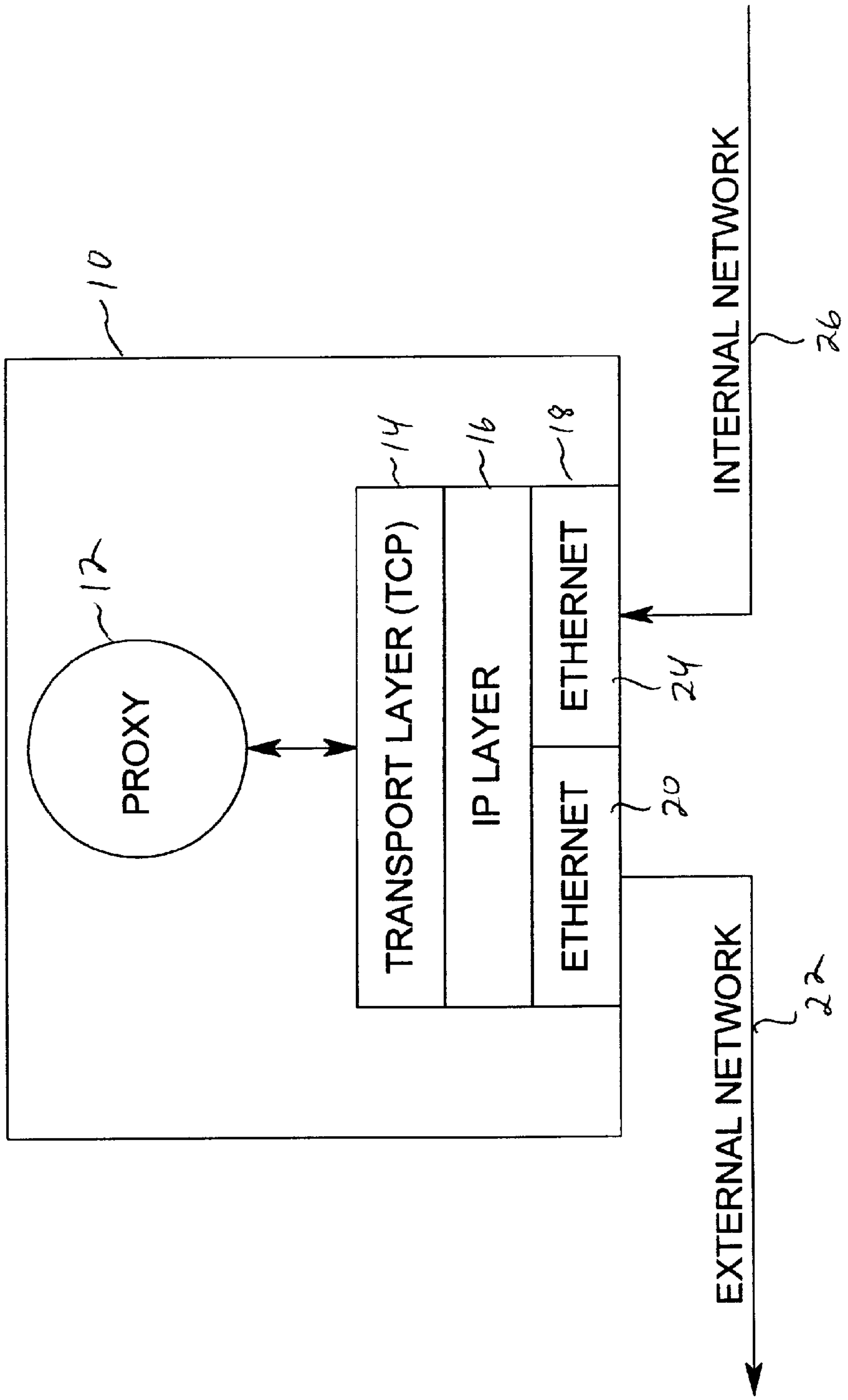

10
PROXY
12
TRANSPORT LAYER (TCP)
14
IP LAYER
16
ETHERNET
18
ETHERNET
20
24
INTERNAL NETWORK
26
EXTERNAL NETWORK
22

Fig. 1

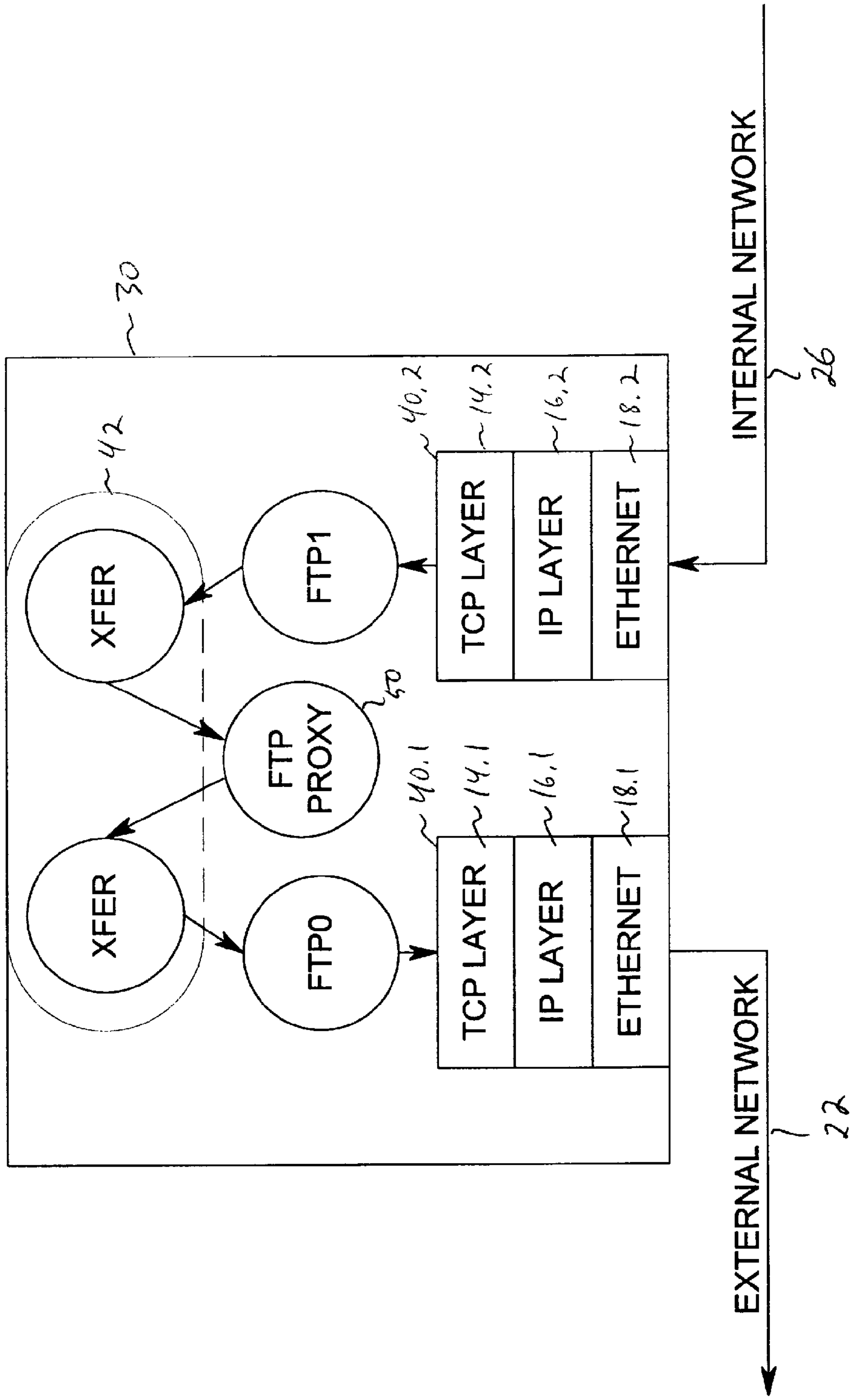

30
42
XFER
FTP1
FTP PROXY
60
XFER
FTP0
40.2
14.2
16.2
18.2
TCP LAYER
IP LAYER
ETHERNET
40.1
14.1
16.1
18.1
TCP LAYER
IP LAYER
ETHERNET
INTERNAL NETWORK
26
EXTERNAL NETWORK
22

Fig. 2

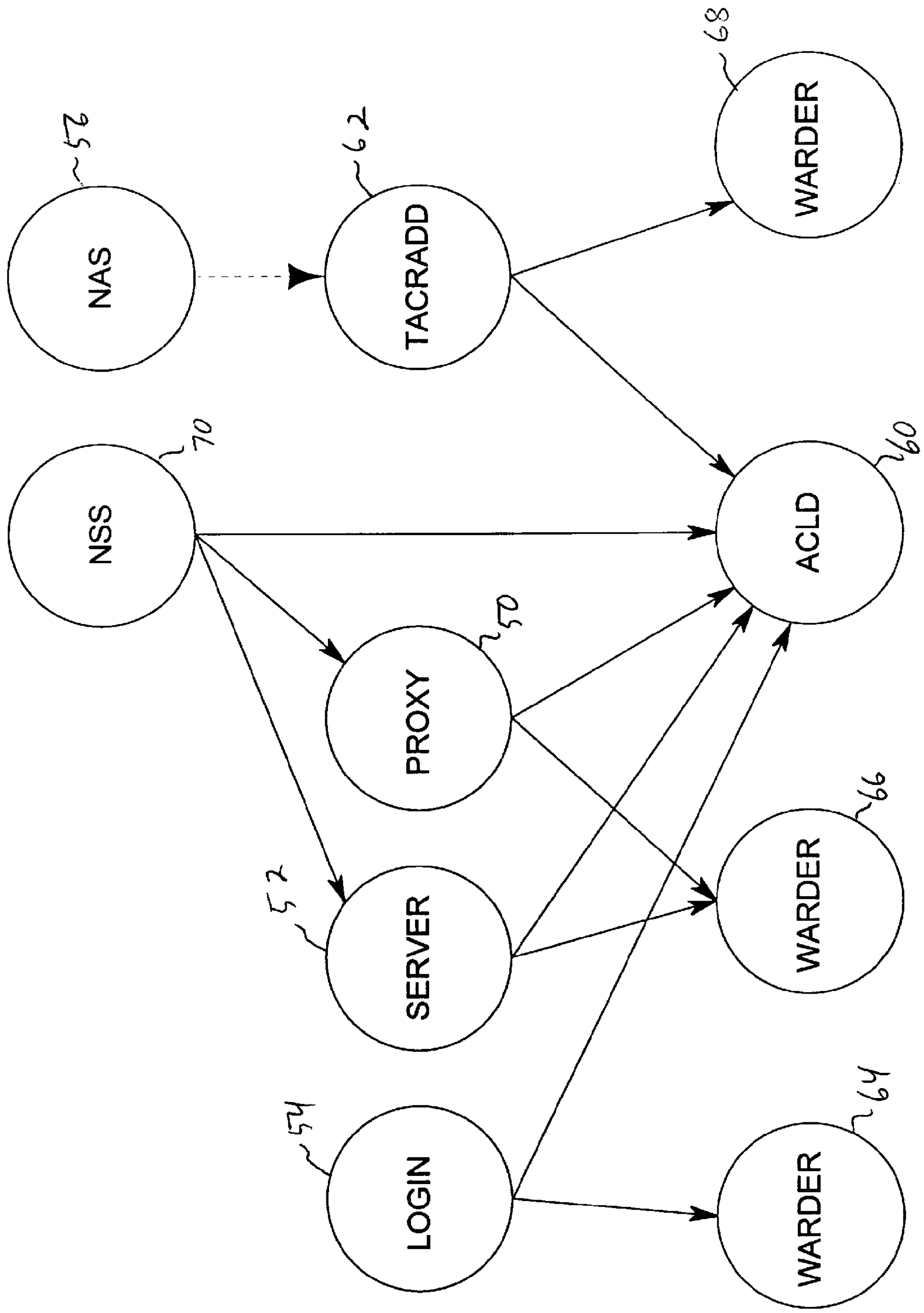

NAS
56
TACRADD
62
WARDER
68
NSS
70
ACLD
60
PROXY
50
SERVER
52
WARDER
66
LOGIN
54
WARDER
64

Fig. 3

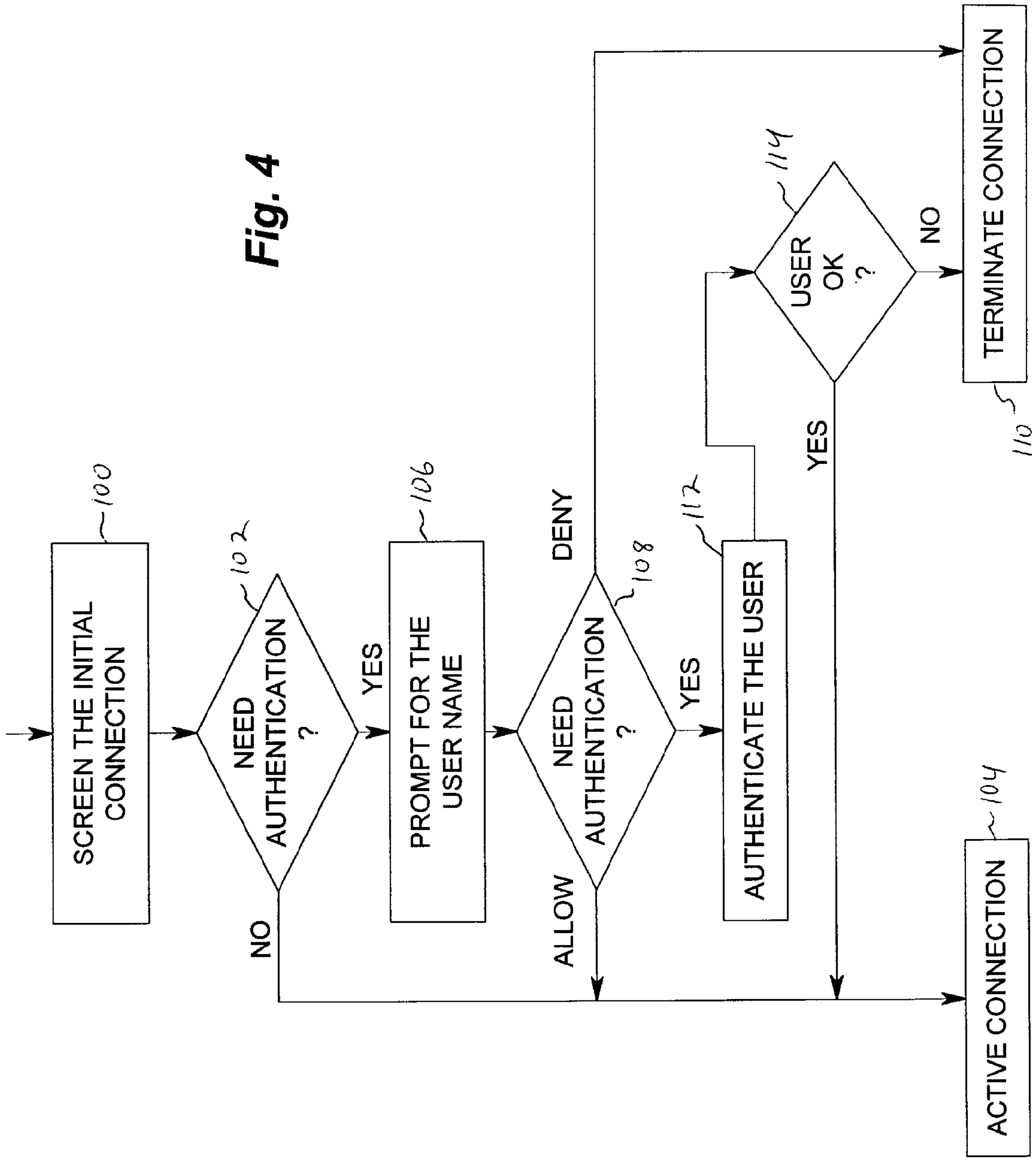
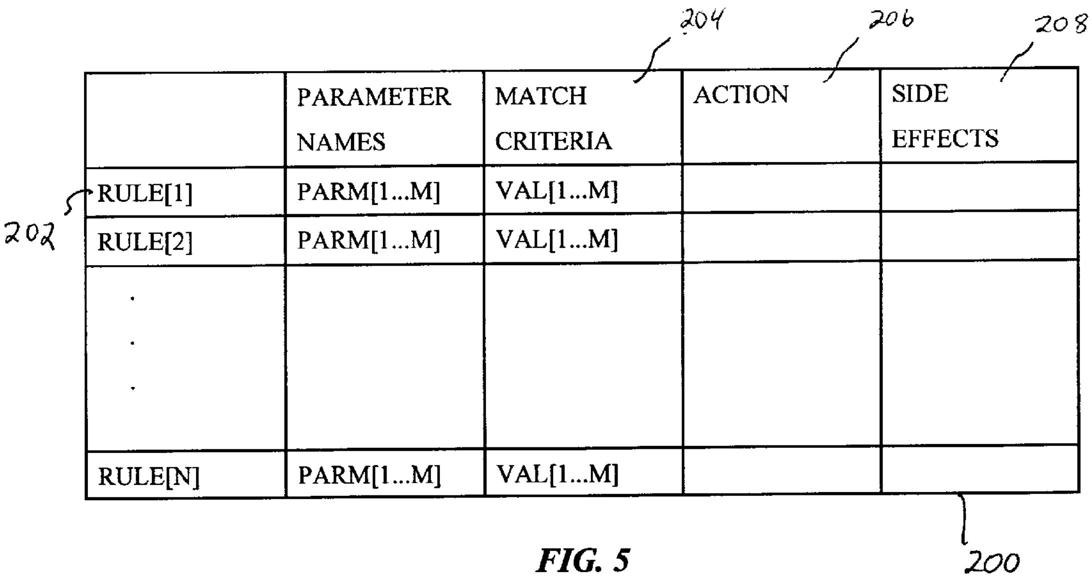

SCREEN THE INITIAL CONNECTION
100
NEED AUTHENTICATION ?
102
YES
NO
PROMPT FOR THE USER NAME
106
NEED AUTHENTICATION ?
108
DENY
ALLOW
YES
AUTHENTICATE THE USER
112
USER OK ?
114
YES
NO
TERMINATE CONNECTION
110
ACTIVE CONNECTION
104

Fig. 4



	PARAMETER NAMES	MATCH CRITERIA	ACTION	SIDE EFFECTS
RULE[1]	PARM[1...M]	VAL[1...M]		
RULE[2]	PARM[1...M]	VAL[1...M]		
. . .				
RULE[N]	PARM[1...M]	VAL[1...M]		

204
206
208
202
200

FIG. 5

GENERALIZED SECURITY POLICY MANAGEMENT SYSTEM AND METHOD

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention pertains generally to network communications, and in particular to a system and method for regulating the flow of internetwork connections through a firewall.

2. Background Information

Firewalls have become an increasingly important part of network design. Firewalls provide protection of valuable resources on a private network while allowing communication and access with systems located on an unprotected network such as the Internet. In addition, they operate to block attacks on a private network arriving from the unprotected network by providing a single connection with limited services. A well designed firewall limits the security problems of an Internet connection to a single firewall computer system. This allows an organization to focus their network security efforts on the definition of the security policy enforced by the firewall. An example of a firewall is given in "SYSTEM AND METHOD FOR PROVIDING SECURE INTERNETWORK SERVICES", U.S. patent application Ser. No. 08/322078, filed Oct. 12, 1994 by Boebert et al., allowed, the description of which is hereby incorporated by reference. A second example of such a system is described in "SYSTEM AND METHOD FOR ACHIEVING NETWORK SEPARATION", U.S. application Ser. No. 08/599,232, filed Feb. 9, 1996 by Gooderum et al., pending, the description of which is hereby incorporated by reference. Both are examples of application level gateways. Application level gateways use proxies operating at the application layer to process traffic through the firewall. As such, they can review not only the message traffic but also message content. In addition, they provide authentication and identification services, access control and auditing.

Access Control Lists, or ACLs, are lists of rules that regulate the flow of Internet connections through a firewall. These rules control how a firewall's servers and proxies will react to connection attempts. When a server or proxy receives an incoming connection, it performs an ACL check on that connection.

An ACL check compares the source and destination IP address of the connection against a list of ACL rules. The rules determine whether the connection is allowed or denied. A rule can also have one or more side effects. A side effect causes the proxy to change its behavior in some fashion. For example, a common side effect is to redirect the destination IP address to an alternate machine.

Sidewinder, Version 2.0, is a firewall which is an example of a system which uses an ACL check to regulate the flow of Internet connections through its firewall. ACLs in Sidewinder 2.0 are stored in a file, /etc/sidewinder/acl.conf. The file is read by all of the servers and proxies on the Sidewinder firewall. A line in the file either allows or denies a connection based on the connections source IP address, destination IP address, and destination port number. Some examples are shown below:

```
allowed_flow( source_addr(net_addr(*.*.*.* 0 internal))
              dest_addr(net_addr(*.*.*.* 0 external))
```

-continued

```
              service (ftp tcp)
              0.0.0.0 0)
```

This rule allows access from any client located in the internal security domain to any ftp server located in the external security domain.

```
allowed_flow( source_addr(net_addr(*.*.*.* 0 internal))
              dest_addr(net_addr(*.*.*.* 0 external))
              service (http tcp)
              0.0.0.0 0)
 denied_flow( source_addr(net_addr(*.*.*.* 0 internal))
              dest_addr(net_addr=(174.252.1.1 0 external))
              service(http tcp)
              0.0.0.0 0)
```

The first rule allows http access from the internal security domain to all Web servers in the external security domain. The second rule denies access to a specific web server located at 174.252.1.1.

```
allowed_flow( source_addr(net_addr(*.*.*.* 0 external))
              dest_addr(net_addr(192.168.1.192 0 external))
              service(nntp tcp)
              172.17.192.48 0)
```

This rule intercepts all incoming connections that go the external side of the local Sidewinder (192.168.1.192) and redirects them to shade.sctc.com (172.17.192.48).

In general, ACL rules used in Sidewinder, Version 2.0, have the following matching criteria:

The source IP address. This can be expressed as a subnet by indicating the number of significant bits in the address.

The source security domain. This is always either "internal" or "external".

The destination IP address.

The destination security domain, again either "internal" or "external".

The service name. The names and protocols of the services are obtained from the file /etc/services. and they have the following two side effects:

Redirect the IP address to a different machine.

Redirect the port number to a different port.

A connection from a specific IP source address to a specific IP destination address is denied unless there is a rule that allows the connection and there is no entry that denies the connection. The order of entries in the list does not matter.

An ACL approach like that used in Sidewinder 2.0 has a number of limitations. For instance, since all ACL rules in that firewall system are specified using only IP addresses, there is no way to specify a host name. A rule can have only one source, one destination and one service; a separate rule is needed for each service and for each workstation. Therefore, to block access to several web sites you need to create a separate rule for each one. Furthermore, a site with five services and 1,000 workstations may need 5,000 rules. This can slow performance.

In addition, the use of static IP addresses creates a problem for a site that uses Microsoft Windows NT Server and DHCP (Dynamic Host Configuration Protocol) with desktop personal computers (PCS). The DHCP server assigns an arbitrary IP address from a pool when each PC boots up. It is impossible to assign an ACL rule to a particular PC because its IP address is not fixed.

In addition, there is no place to store a user name. The granularity of access control is on a per-host basis.

Sidewinder 2.0 stores a complete copy of the full access control list in the memory of every proxy. If the number of rules is large, the memory consumed hurts performance. In addition, there is no support for activating rules during certain times of the day or during certain days of the week.

Finally, there is no way to specify a different authentication method for a given connection. For a given service, the authentication method must be the same for all users and for all hosts.

What is needed is a generalized security policy management system which can operate free of these limitations.

SUMMARY OF THE INVENTION

The present invention is a system and method for regulating the flow of internetwork connections through a firewall having a network protocol stack which includes an Internet Protocol (IP) layer. A determination is made of the parameters characteristic of a connection request, including a netelement parameter characteristic of where the connection request came from. A query is generated and a determination is made whether there is a rule corresponding to that query. If there is a rule corresponding to the query, a determination is made whether authentication is required by the rule. If authentication is required by the rule, an authentication protocol is activated and the connection is activated if the authentication protocol is completed successfully.

According to another aspect of the present invention, a system and method of regulating the flow of internetwork connections through a firewall having a network protocol stack which includes an Internet Protocol (IP) layer. An access control list is formed, wherein the access control list includes a plurality of rules, where each rule includes a plurality of rule parameters and values associated with said rule parameters. Query parameters characteristic of a connection request are determined, wherein said query parameters include a netelement parameter characteristic of where the connection request came from and a query is generated which lists the query parameters. The query is then applied to the access control list and a determination is made as to whether there is a rule corresponding to the query. If there is a rule, a determination is made as to whether authentication is required by the rule and, if authentication is required by the rule, an authentication protocol is executed. If the authentication protocol is completed successfully, the connection is activated.

According to yet another aspect of the invention, a network-separated application level gateway firewall is described.

BRIEF DESCRIPTION OF THE DRAWINGS

In the drawings, where like numerals refer to like components throughout the several views:

FIG. **1** is a functional block diagram of an application level gateway firewall according to the present invention;

FIG. **2** is a functional block diagram of a network-separated application level gateway firewall according to the present invention;

FIG. **3** is a block diagram representing the interaction of agents with warders and with the access control list daemon;

FIG. **4** is a representation of the steps an agent goes through in authenticating a connection; and

FIG. **5** is a representation of one embodiment of a ruleset which can be used with the access control list daemon of FIG. **3**.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the following detailed description of the preferred embodiment, references made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific preferred embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention, and it is to be understood that other embodiments may be utilized and that structural, logical, physical, architectural, and electrical changes may be made without departing from the spirit and scope of the present invention. The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the present invention is defined only by the appended claims and their equivalents.

A system **10** which can be used for firewall-to-firewall encryption (FFE) is shown in FIG. **1**. In FIG. **1**, system **10** includes a workstation **12** communicating through a firewall **14** to an unprotected network **16** such as the Internet. System **10** also includes a workstation **20** communicating through a firewall **18** to unprotected network **16**. In one embodiment, firewall **18** is an application level gateway.

A firewall which can be used to regulate the flow of internetwork connections from an internal to an external network is shown in FIG. **1**. Firewall **10** is an application level gateway. As noted above, application level gateways use proxies **12** operating at the application layer to process traffic through the firewall. As such, they can review not only the message traffic but also message content. In addition, they provide authentication and identification services, access control and auditing. As can be seen in FIG. **1**, proxy **12** is connected through transport layer **14** and Internet Protocol (IP) layer **16** to a physical layer **18**. Physical layer **18** includes an Ethernet connection **20** to an external network **22** and an Ethernet connection **24** to an internal network **26**.

A network separation version of a firewall is shown in FIG. **2**. In FIG. **2**, network separation is used to divide firewall **30** into a set of two independent regions or burbs, with a domain and a protocol stack assigned to each burb. Each protocol stack **40** has its own independent set of data structures, including routing information and protocol information. A given socket will be bound to a single protocol stack at creation time and no data can pass between protocol stacks **40** without going through proxy space **42**. A proxy **50** therefore acts as the go-between for transfers between domains. Because of this, a malicious attacker who gains control of one of the regions is prevented from being able to compromise processes executing in other regions.

Network separation and its application to an application level gateway is described in "SYSTEM AND METHOD FOR ACHIEVING NETWORK SEPARATION", U.S. application Ser. No. 08/599,232, filed Feb. 9, 1996 by Gooderum et al., pending, the description of which is hereby incorporated by reference. Network separation does provide some significant advantages in implementing an access control scheme. For instance, a school may want to put student machines behind an outer Internet firewall and put the teacher machines behind an inner firewall (in order to protect the teachers' machines from the students, e.g., so that the students don't tamper the teachers' grading files). One solution is to install two firewall systems. A cheaper solution is to provide a single firewall **30** that has three burbs: "external", "student" and "teacher", with ACLs that restrict flow from the outer rings to the inner rings. In other words, multiple burbs allow nested levels of protection without the expense of acquiring additional firewalls.

Furthermore, encryption can be used with network separation to multiplex burbs on a single physical layer interface **24**. For example, two burbs could be assigned to a single Ethernet interface card. Each burb has its own network protocol stack; each stack, however, is connected to the same interface card. By encrypting one channel, leaving the other channel unencrypted and including in the ACL information as to whether a particular message is encrypted or unencrypted, it is possible for both burbs to use the same Ethernet card.

As noted above, an Access Control List, or ACL, is a list of rules that regulate the flow of Internet connections through a firewall. These rules control how a firewall's servers and proxies will react to connection attempts. When a server or proxy receives an incoming connection, it performs an ACL check on that connection.

An ACL check compares a set of parameters associated with the connection against a list of ACL rules. The rules determine whether the connection is allowed or denied. A rule can also have one or more side effects. A side effect causes the proxy to change its behavior in some fashion. For example, a common side effect is to redirect the destination IP address to an alternate machine. In addition to IP connection attempts, ACL checks can also made on the console logins and on logins made from serial ports. Finally, ACL checks can also be made on behalf of IP access devices, such as a Cisco box, through the use of the industry standard TACACS+ protocol.

In one embodiment, the ACL is managed by an acld daemon running in the kernel of firewalls **10** and **30**. The acld daemon receives two types of requests, one to query the ACL and one to administer it. In one such embodiment, the ACL is stored in a relational database such as the Oracle database for fast access. By using such a database, query execution is asynchronous and many queries can be executing concurrently. In addition, these types of databases are designed to manipulate long lists of rules quickly and efficiently. These qualities ensure that a given query cannot hang up the process that issued the query for any appreciable time (>1–2 seconds).

The current database can hold up to 100,000 users and up to 10,000 hosts but can be scaled up to the capacity of the underlying database engine. The results of an ACL check is cached, allowing repeated checks to be turned around very quickly.

Applications on firewalls **10** and **30** can query acld to determine if a given connection attempt should be allowed to succeed. In one embodiment, such as is shown in FIG. **3**, the types of applications (i.e. "agents") that can make ACL queries can be divided into four classes:

1) Proxies **50**. These allow connections to pass through firewall **10** or **30** in order to provide access to a remote service. They include tnauthp (authenticated telnet proxy), pftp (FTP proxy), httpp (HTTP proxy), and tcpgsp (TCP generic service proxy).
2) Servers **52**. These provide a service on the firewall itself. They include ftpd and httpd.
3) Login agents **54**. Login agent **54** is a program on the firewall that can create a Unix shell. It is not considered a server because it cannot receive IP connections. One example is /usr/bin/login when used to create a dialup session or a console session on firewall **10** or **30**. Another example is the command srole.
4) Network Access Servers (NAS) **56**. NAS **56** is a remote IP access device, typically a dialup box manufactured by such companies as Cisco or Bridge. The NAS usually provides dialup telnet service and may also provide SLIP or PPP service.

Proxies **50**, servers **52**, login agents **54**, and NASes **58** make queries to acld **60** to determine if a given connection attempt should be allowed to succeed. All of the agents except NAS **58** make their queries directly. NAS **58**, because it is remote, must communicate via an auxiliary daemon that typically uses an industry standard protocol such as RADIUS or TACACS+. The auxiliary daemon (e.g., tacradd **62**) in turn forwards the query to local acld **60**.

As a side effect of the query, acld **60** tells the agent if authentication is needed. If no authentication is needed, the connection proceeds immediately. Otherwise acld **60** provides (as another side effect) a list of allowed authentication methods that the user can choose from. The agent can present a menu of choices or simply pick the first authentication method by default. Typical authentication methods include plain password, SNK DSS, SDI SecurID, LOCKout DES, and LOCKout FORTEZZA. In one embodiment, the list of allowed authentication methods varies depending on the host name, user name, time of day, or any combination thereof.

In the embodiment shown in FIG. **3** authentication is actually performed by a special server called a warder. The agent contacts the warder to perform authentication. In one embodiment, a warder is associated with each authentication method. For instance, warder **64** may be the SNK DSS warder, warder **66** may be the LOCKout DES warder and warder **68** may be the LOCKout FORTEZZA warder. A warder is a process which performs an authentication method. In one embodiment, all warders have a common, standard, authentications interface used to talk to the agents. By using this warder structure, a single agent can therefore access any of the available authentication methods simply by contacting the appropriate warder. There is no direct communication between the warder and acld.

A representative proxy agent **50** is described in VIRTUAL PRIVATE NETWORK ON APPLICATION GATEWAY, U.S. patent application Ser. No. 08/715,343, filed herewith, the description of which is hereby incorporated by reference.

The system administrator is responsible for creating the ACL ruleset. The ACL ruleset is a reflection of the administrator's security policy. The administrator needs to answer the following questions before creating the ruleset:

1) Where are the boundaries of the security domains?
2) Which security domains should be allowed to initiate connections into other security domains?
3) What types of information should be allowed to flow between the security domains?
4) Should connections be allowed based on host name, user name, time of day, or some combination thereof?
5) What type(s) of user authentication, if any, should be required to enter a security domain?

In the preferred embodiment, the ACL ruleset can be modified by the administrator using either a graphical user interface, or GUI, or a command line interface. Changes to the ACL ruleset are stored in the internal database. To prevent the occurrence of transient states that may violate the security policy, the database is locked during periods in which the administrator is making more than one rule update. (This is especially important when reloading the entire database from diskette or tape.)

In one embodiment, the database is implemented in SQL using either Faircom's C-Tree or Just Logic's SQL Database Manager. In another embodiment, acld **60** is connected to an external client/server commercial SQL database engine, such as Oracle. This will give the customer additional flexibility in administering the ruleset.

In one embodiment non-time-critical portions of the ACL control and administration code is written in Python. Time-critical portions are coded in C.

Generating a Query

To make an ACL check, the agent collects information about the nature of the connection. This information includes the source and destination IP address. The agent places this information into a query list. The query list contains all of the relevant information needed to make the ACL check. The agent then submits the query list to acld **60** and acld **60** searches for a rule that matches the query list and returns a reply list. The reply list includes either "allow" or "deny" to indicate if the connection should be accepted or rejected. Other values in the reply list are side effects that change the behavior of the agent.

In one embodiment the query list contains the following pieces of information:

The source IP address, if applicable.

The destination IP address, if applicable.

The source security domain.

The destination security domain.

The network protocol (TCP or UDP), if applicable.

The type of agent (proxy, server, login, or NAS).

The type of IP encryption used by the connection, if any.

The name of the user (when known).

The name of the user's selected warder (when known).

This information is used by acld **60** to search for a rule that "best" matches the query.

Rule Precedence

In general, rules with specific values for a query parameter are preferred over rules with a wildcard parameter. For example, a rule that says user="alan" is preferred over a rule that says user=* (where * is a wildcard character). Sometimes it is hard, however, to decide which rule is best. For example, one rule may say host="t-bone", user=* while another rule may say host=*, user="alan". Which rule is "best" in this case? The answer is "it depends". The system administrator needs to decide which is more important: hosts or users, and embed the security policy in the ACL accordingly.

In one embodiment, ACL rules are checked in sequential order. The first rule that matches the query is chosen. In one such embodiment, when the administrator creates a new rule, the acld interface suggests a position in the rule set based on a predefined precedence scheme. The position of the rule is the sole criteria for determining whether or not the rule is selected. The administrator can override the suggested position if he or she so desires and the interface will perform a sanity check and warn the administrator if a rule's position is obviously wrong.

For example, assume there are two rules 'deny_all', and 'allow_alan':

Name:	deny_all	allow_alan
Position:	1	2
Action:	deny	allow
User(s) :	*	alan

In this example the acld interface would flag a warning that the second rule (allow_alan) will never match a query because the first rule (deny_all) has a wildcard for the user name.

It should be obvious that other precedence schemes could be used, including schemes which place a higher security value on the host rather than the user, and vice versa.

Once a rule is selected, the agent receives a reply list. The reply can allow or deny the connection (if the agent sets a flag a verbose explanation is sent explaining why the connection was allowed or denied. For each rule in the ruleset, the reply returns a string that explains the result of the comparison with the query list. If the rule did (not) match the query, it returns a string that explains why (not).

The reply can demand authentication. If authentication is demanded, the reply includes a list of allowed warders. The agent can present a menu of authentication method choices to the user or may simply pick the first warder as the default.

The reply can tell the agent to redirect the destination IP address to a different machine. This is only applicable to proxies **50**.

The reply can tell the agent to redirect the destination port number to a different port. This is only applicable to proxies **50**.

In one embodiment each rule has a name and the reply returns the name of the rule that matched the query. This is useful for troubleshooting problems with the ruleset. In another embodiment the reply returns the position of the rule in the ruleset. Again this is useful for troubleshooting.

The reply can provide additional side effects that are unique to the service. For example, for an FTP service it determines whether GET or PUT operations are allowed. For an HTTP service it determines what types of URLs are blocked by the HTTP proxy.

The reply can include an expiration date for the result of this query. This is used internally for caching. If a duplicate query is made by the same agent before the time expires, the cached reply is returned.

The agent should examine the values in the reply list and act upon them appropriately. Though not strictly required to do so, the agent is expected to abide by the results of the query.

ACL Check Procedure

The steps followed in executing an ACL check are shown in FIG. **4**. In FIG. **4**, the process starts at **100** where an agent can perform an optional initial ACL check when a connection is first detected. Most agents will perform an initial ACL check when a connection is first detected. The reason is that connections from some hosts are always disallowed. In this case the connection should be rejected with no further ado. For most proxies and servers, Network Services Sentry (NSS) **70** will screen out the initial connection.

Note: If the agent is incapable of doing authentication (e.g., gopher or WAIS), NSS **70** should set the selected warder to "none" before it does the initial check. This avoids a potential ambiguity created by a tentative check on an unknown user name (explained later).

Some agents, such as httpd, listen for connections directly and do not depend on NSS. These agents have to make the initial check themselves.

If the reply to the initial check says "deny" the connection should be closed at that point and no further communication should occur. If the initial check says "allow" the agent can proceed to step **102**. As we shall see below, "allow" really means "maybe" until further information about the user is obtained.

If the reply to the initial check says "allow," at **102** the agent should examine the reply to see if it demands authentication of the user. Note that if the agent was screened by NSS **70**, the agent will have to issue a duplicate check to get the ACL information because NSS **70** does not pass the ACL information to the agent.

If the ACL reply does not demand authentication, the ACL check procedure is complete and the agent can proceed to **104** to open the connection. Note that some servers, such as ftpd, will always demand authentication anyway.

If, however, the ACL reply demands authentication, the agent should proceed to **106** and prompt for the user name. Note that some services such as gopher and WAIS do not provide a means to ask for a user name. In this case acld would have rejected the connection at the initial ACL check (because NSS **70** set the warder name to "none").

In one embodiment, a "magic window" is opened outside of the regular service to authenticate services like gopher and WAIS. In one such embodiment, a successful authentication will open a timed window to those services to allow access.

The prompt for a user name should include a way to specify the name of the warder. For example:

login: alan
login: alan:securid
login: alan:snk
login: alan:lockout
login: alan:fortezza
login: alan:password

Once the agent knows the user name, it should move to **108** and do a second ACL check. The query parameters in the second check should include the same parameters as the first check plus the name of the user. It should also include the name of the selected warder (if the user specified one).

If the reply to the second check says "deny" the agent should move to **110** and drop the connection. The agent may, however, want to first issue a dummy password/challenge prompt to avoid leaking information.

If the reply to the check says "allow," the reply parameters will include a list of allowed warders for that user. The agent should make sure that the user's selected warder is in the list of allowed warders. (If the agent passed the name of the selected warder to acld **60**, this test is done automatically by acld **60**.)

If the user did not select a warder, the agent should pick the first warder in the list of allowed warders and proceed to **112** to authenticate the user. At this point the agent is done making ACL checks. (In one embodiment, a menu of available warders is displayed for the user and the user selects one of the warders from the list of available warders. In such an embodiment, however, in order to prevent information leakage all warders should be listed, even those that do not actually apply.)

At **112**, the agent authenticates the connection with the selected warder. The agent should contact the selected warder and perform the authentication. This may include a challenge/response sequence. Please note that if the user changes his/her login name while talking to the warder, the agent must recheck the ACL with the new user name.

A check of the results of the authentication is made at **114** and, if at **114** the user passes the authentication check, the agent proceeds to **104**. If, however, the user fails the authentication check, the agent proceeds to **110** and drops the connection.

Time Intervals

In one embodiment, ACL rules can be configured to specify a time interval. If a connection is active when the time interval expires, acld **60** will notify each agent to reassess the connection. In one embodiment, when a time interval expires each agent receives an asychronous message on the acld socket. The message tells the agent to recheck all of its active connections.

The agent should keep a saved copy of the query list for every active connection. When it receives notification that a time interval has expired, it should reissue ACL checks for all the saved query lists. (The agent does not need to reauthenticate.) If the reply to an ACL check is "deny," the agent should drop the corresponding connection. If the agent wants to be polite, it can send a warning message to the user and provide a grace period, allowing the user to clean up.

A similar asynchronous notification is also sent by acld **60** to all agents whenever the administrator changes the ACL ruleset.

The ACL Rule

The heart of the ACL system is the rule. The ACL database contains a list of rules, called the ruleset. A representative ruleset **200** is shown in FIG. **5**, where a rule **202** contains three types of attributes: match criteria **204**, the action **206**, and side effects **208**.

When an agent submits a query list, the parameters in the query list are compared against the match criteria **204** in each rule **202**.1 through **202**.N. The first rule **202** that matches all of the criteria is returned (except when making a tentative check, which can match more than one rule.)

In one embodiment, the match criteria of a rule is as follows:

1) The source netelement. A netelement is a host name, a subnet name, a domain name, an IP address, or a netgroup name. (A netgroup contains netelements.) If omitted, the source is wildcarded. By using netelements and netgroups, it is possible to name groups of machines symbolically, and also to create groups of groups (i.e., the netgroup).
2) The destination netelement. If omitted, the destination is wildcarded.
3) The source security domain. If omitted, the domain is wildcarded.
4) The destination security domain. If omitted, the domain is wildcarded.
5) A list of agent types: proxy, server, login, and/or NAS.
6) A list of service names. A service name is usually a name found in /etc/services such as "ftp" or "http". For a login agent, the service name can also be one of the following:
   a) console: a login from the system console.
   b) deal: a login from a serial dialup port.
   c) telnet: a login from a telnet connection.
7) The network protocol: either TCP or UDP. The default is TCP.
8) The minimum encryption required. This can be either none or IPSEC. The connection is rejected if the encryption level indicated in the query is not as strong as that of the rule.
9) The name of the usergroup. If omitted, the usergroup is wildcarded.
10) A list of allowed warders. The list of allowed warders is returned to the agent in order to present a menu of choices to the user. If the user specifies a warder, it must match one in this list. Otherwise the connection is rejected.
11) A list of time intervals during which the rule is active.
12) An "ignore" flag. If this flag is set, the rule is ignored. This can be used to temporarily disable a rule without deleting it.

Other embodiments include subsets of the above match criteria. In one embodiment, it is possible to assign a collection of users, hosts or services to a symbolic name. This is demonstrated above for groups of machines and for groups of users but also could be used to represent services. By doing this one could arrive at a single ACL rule **202** covering the majority of users, services and machines. For instance, a single rule **202** may have services=Standard_ services and users=Internet_users. To enable service for a new employee Joe Smith, his name would simply be added to the list of users in the Internet_users group; a new rule **202** would not be necessary.

The most important attribute of a rule is the action. In one embodiment, each action can assume one of two values: allow or deny. Allow means to accept the incoming connection. Deny means to reject the incoming connection.

In another embodiment, each action can assume one of three values: allow, deny or default. In this embodiment default is a special action that provides the ability to specify common side effects for multiple rules; on receiving a query acld **60** collects all the side effects from the matching default rules and merges them with the final allow or deny rule. Merging happens in sequential order of the rules, with the side effects of the final rule overriding the side effects of the default rule(s).

The side effects of a reply can change the behavior of the agent in some way. For instance, the reply can demand authentication. Each reply contains a flag indicating if authentication is needed. If set, the agent must authenticate the user. It must prompt for a user name, then select a warder, and then contact the selected warder.

If authentication is needed, the reply includes a list of allowed warders. The list of allowed warders allows the agent to present a menu of choices to the user. If the user specifies a warder, the agent must verify that it matches one of the choices in this list.

The reply can tell the agent to redirect the destination IP address to a different machine. This is only applicable to proxies **50**.

The reply can tell the agent to redirect the destination port number to a different port. This is only applicable to proxies **50**.

The reply can provide additional side effects that are unique to the service. These are called service parameters. For example, for FTP the service parameters indicate whether GET or PUT operations are allowed. For HTTP the service parameters indicate the types of URLs that are blocked.

As noted previously, in one embodiment the ACL is implemented in a relational database. Such an implementation carries the advantage that the ACL is extensible; new parameters can be defined without redoing the entire ruleset.

To help give you a better understanding of how ACL checks work, this section presents several examples. Let's start with a simple example, a ruleset that contains only one rule:

Name:	telnet_out
Position:	1
Action:	allow
Ignore:	no
Source:	*
Dest:	*
Source Sec Domain:	internal
Dest Sec Domain:	external
Agents:	[proxy]
Services:	[telnet]
Protocol:	tcp
Usergroup:	*
Time Intervals:	[]
Redir Host:	
Redir Port:	
Auth Needed:	no
Min Encrypt:	none
Alert:	none

-continued

Allowed Auth Methods:	[]
Service Parameters:	{}
Comments:	""

The rule allows any client located in the internal security domain to connect to any telnet server located in the external security domain. No authentication is required.

Here is a ruleset with two rules:

Name:	ftp_out	ftp-in
Position:	1	2
Action:	allow	allow
ignore:	no	no
Source:	*	*
Dest:	*	local
source Sec Domain:	internal	external
Dest Sec Domain:	external	external
Agents:	[proxy]	[server]
Services: [ftp]	[ftp]	
Protocol:	tcp	tcp
usergroup:	*	Anonymous
Time Intervals:	[]	["Sat-sun", "Mon mid-8am", "Mon-Fri 5pm-mid")
Redir Host:		
Redir Port:		
Auth Needed:	no	yes
Min Encrypt:	none	none
Alert:	none	none
Allowed Auth Methods:	[]	[pas]
Service Parameters:	{}	{ftp:[get]}
Comments:	'anonymous FTP is allowed outside of business hours'	

The first rule (ftp_out) allows any client in the internal security domain to access any FTP server in the external domain. The second rule supports an anonymous FTP server on the local firewall (local always refers to the local firewall regardless of the security domain). Access to the server is limited to outside of business hours and only GET is allowed.

Here is a ruleset for an organization with a traveling sales force. Some salespersons have laptops with IP encryption software.

Name:	out	sales-crypt	sales-nocrypt
Position:	1	2	3
Action:	allow	allow	allow
Ignore:	no	no	no
Source:	*	*	*
Dest:	*	*	*
Source Sec Domain:	internal	external	external
Dest Sec Domain:	external	internal	internal
Agents:	[proxy]	[proxy]	[proxy]
Services:	[telnet, ftp]	[telnet, ftp]	[telnet, ftp]
Protocol:	tcp	tcp	tcp
Usergroup:	*	Sales	Sales
Time Intervals:	[]	[]	[]
Redir Host:			
Redir Port:			
Auth Needed:	no	yes	yes
Min Encrypt:	none	ipsec	none
Alert:	none	none	none
Allowed Auth	[]	[passwd,securid]	[securid]

-continued

Methods:			
Service Parameters:	{}	{ftp: [get,put]}	{ftp: [get,put]}
Comments:		"encryption plain pas ok"	"no encryption strong auth required"

Weaker authentication is permitted if IP encryption is used, because the plain passwords (pas) cannot be observed by packet sniffers.

Below is an example of why some ACL checks cannot be fully resolved until the user name is known.

Name:	Other	Sales	Eng
Position:	1	2	3
Action:	deny	allow	allow
ignore:	no	no	no
Source:	*	*	*
Dest:	domain sctc.com	domain sctc.com	domain sctc.com
Source Sec Domain:	external	external	external
Dest Sec Domain:	internal	internal	internal
Agents:	[proxy,nas]	[proxy, nas]	[proxy,nas]
Services:	[telnet,ftp]	[telnet,ftp]	[telnet,ftp]
Protocol:	tcp	tcp	tcp
Usergroup:	Other	Sales	Eng
Time Intervals:	[]	[]	[]
Redir Host:			
Redir Port;			
Auth Needed:	n/a	yes	yes
Min Encrypt:	n/a	none	none
Alert:	n/a	none	none
Allowed Auth Methods:	n/a	[securid]	[lockout]
Service Parameters:	n/a	{ftp: [get])	{ftp: [get]}
Comments:	"Deny users\ in the Other\ category'	"sales"	"engineers"

When the initial ACL check is made at **102**, the name of the user is not yet known. Therefore all three rules can potentially match. In a situation like this acld **60** performs a tentative check. A tentative check will return "allow" if there exists at least one rule that might allow the connection, unless a deny rule exists that will certainly reject the connection. (For example if the first rule had a wildcard for the usergroup instead of Other, acld **60** would reject the connection outright.)

A tentative check can match more than one "allow" rule. When this happens, acld **60** will merge all of the allowed warders together. For the above example it will return the following side effects:

Auth Needed:	yes	
Allowed Auth Method:	[securid, lockout]	Both

Both SecurID and LOCKout are returned as allowed warders. In general, acld **60** will return the union of all allowed warders for all tentatively matching rules. In the embodiment shown the default warder is SecurID because Sales comes before Eng.

When the agent makes its second check at **108** including the user name, it will match exactly one rule and will return only the one warder name. This is called a final check. Knowledge of the user name is crucial to selecting the final rule. If the agent is incapable of doing authentication (e.g, gopher or WAIS), it should set the selected warder to "none". The warder named "none" is special because it forces acld **60** to do a final check instead of a tentative check.

In summary, there are two ways to force acld **60** to do a final check:

1. Specify the user name in the query list; or
2. Set the selected warder to "none" in the query list.

Otherwise acld **60** will do a tentative check and may match more than one rule. The agent is responsible for doing a final check on the connection once the user name is known. Failing to do a final check is a serious error.

In one embodiment ruleset **200** includes a URL filter parameter. One of the values included in an ACL query could then be the URL that the user is trying to access. Ruleset **200** would then include entries for particular URLs or groups of URLs that have been banned or restricted. In one embodiment, a rating service such as WebTrack™, available from Webster Network Strategies™ could be used to filter URLs. In WebTrack™, URLs are grouped into categories based on hate speech, sexually explicit material, etc. The ACL ruleset could then be used to restrict or ban access to categories of URLs.

To summarize, during the life of a TCP connection, the ACL database should be queried at four different times:

1) by nss **70** when the TCP connection is first attempted. The name of the user is unknown at this point so acld **60** makes a 'tentative check'. A tentative check will succeed if an allow rule is found that accepts at least one user given the src, dest, service, etc. It will fail if a deny rute is found with a wildcard usergroup.
2) by the agent (proxy **50**, server **52** or login agent **54**) if it must know whether or not to prompt for a user name. Some proxies will always prompt for a user name and therefore will not need to make this check.
3) by the agent after it gets the user name and (optionally) the name of the warder. At this point the agent has all of the information necessary to perform a final check. As a side effect, acld **60** will return a list of allowed warders, the first of which is the default warder. The agent should verify that the selected warder is in the list of allowed warders. The agent can then do authentication with the selected warder. Note that the list can be empty, which means that no authentication is required.
4) by the agent when acld **60** sends an async notification that indicates that ACL ruleset **200** was changed by the administrator. The agent should recheck all of its active connections. A notification is also sent whenever a time-based rule crosses a time boundary.

Although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that any arrangement which is calculated to achieve the same purpose may be substituted for the specific embodiment shown. This application is intended to cover any adaptations or variations of the present invention. Therefore, it is intended that this invention be limited only by the claims and the equivalents thereof.

What is claimed is:

1. A method of regulating the flow of internetwork connections through a firewall having a network protocol stack, wherein the network protocol stack includes an Internet Protocol (IP) layer, the method comprising the steps of:

determining parameters characteristic of a connection request, wherein the parameters include a netelement parameter characteristic of where the connection request came from;

generating a query, wherein the step of generating a query includes the step of adding the parameters to a query list;

determining if there is a rule corresponding to the query;

if there is a rule, determining if authentication is required by the rule;

if authentication is required by the rule, executing an authentication protocol; and

activating the connection if the authentication protocol is completed successfully.

2. The method according to claim **1** wherein the step of executing an authentication protocol includes the step of calling a warder.

3. The method according to claim **1** wherein the step of determining if there is a rule corresponding to the query includes the step of accessing a relational database with the query.

4. The method according to claim **1** wherein the netelement parameter identifies a group of host names.

5. The method according to claim **1** wherein the netelement parameter identifies a group of IP addresses.

6. The method according to claim **1** wherein the netelement parameter identifies a group of subnets.

7. The method according to claim **1** wherein the netelement parameter identifies a group of netgroups, wherein a netgroup is a collection of netelements.

8. The method according to claim **1** wherein the netelement parameter identifies a group of DNS domains.

9. A method of regulating the flow of internetwork connections through a firewall having a network protocol stack, wherein the network protocol stack includes an Internet Protocol (IP) layer, the method comprising the steps of:

forming an access control list, wherein the access control list includes a plurality of rules, wherein each rule includes a plurality of rule parameters and values associated with said rule parameters;

determining query parameters characteristic of a connection request, wherein said query parameters include a netelement parameter characteristic of where the connection request came from;

generating a query, wherein the step of generating a query includes the step of adding the query parameters to a query list;

applying the query to the access control list, wherein the step of applying includes the step of determining if there is a rule corresponding to the query;

if there is a rule, determining if authentication is required by the rule;

if authentication is required by the rule, executing an authentication protocol; and

activating the connection is the authentication protocol is completed successfully.

10. The method according to claim **9**, wherein the method further comprises the steps of:

monitoring time of day; and

closing connections at predefined times of the day.

11. The method according to claim **9**, wherein the query parameters include a URL parameter.

12. A firewall, comprising:

a first communications interface;

a second communications interface;

a first network protocol stack connected to the first communications interface, wherein the first network protocol stack includes an Internet Protocol (IP) layer and a transport layer;

a second network protocol stack connected to the second communications interface, wherein the second network protocol stack includes an Internet Protocol (IP) layer and a transport layer, wherein communication between the first and second communications interfaces passes through a proxy operably coupled to the protocol stacks and communication is otherwise restricted between the protocol stacks;

an access control list process, wherein the access control list process accesses a plurality of rules implementing a security policy; and

an agent, connected to the access control list process and to the transport layers of said first and second network protocol stacks, wherein the agent receives messages from the transport layer, sends the access control list process a query based on parameters associated with the message and executes an authentication protocol selected by the access control list process as a result of the query.

13. The firewall according to claim **12** wherein the first network protocol stack and the first communications interface are assigned to a first burb and the second network protocol stack and the second communications interface are assigned to a second burb.

14. The firewall according to claim **13** wherein the query includes a burb symbol identifying the first burb and wherein the authentication protocol selected by the access control list process varies as a function of the burb symbol.

15. The firewall according to claim **12** wherein the query includes a sender symbol identifying the sender of the message and wherein the authentication protocol selected by the access control list process varies as a function of the sender symbol.

16. The firewall according to claim **12** wherein the first network protocol stack includes a decryption process, operating at the IP layer, that decrypts encrypted messages received by said first communications interface and forwards the decrypted message to the agent via the transport layer.

17. The firewall according to claim **12** wherein the authentication protocol selected by the access control list process varies as a function of whether a message was encrypted or not when received by the IP layer.

18. A computer program product, comprising:

a computer usable medium having computer readable program code embodied thereon, the computer readable program code, when executed, implementing on the computer a method of regulating the flow of internetwork connections through a firewall having a network protocol stack, wherein the network protocol stack includes an Internet Protocol (IP) layer, the method comprising the steps of:

determining parameters characteristic of a connection request, wherein the parameters include a netelement parameter characteristic of where the connection request came from;

generating a query, wherein the step of generating a query includes the step of adding the parameters to a query list;

determining if there is a rule corresponding to the query;

if there is a rule, determining if authentication is required by the rule;

if authentication is required by the rule, executing an authentication protocol; and

activating the connection if the authentication protocol is completed successfully.

19. The computer program product according to claim **18**, wherein the step of executing an authentication protocol includes the step of calling a warder.

* * * * *

US005983350A

United States Patent [19]

Minear et al.

[11] **Patent Number: 5,983,350**

[45] **Date of Patent: *Nov. 9, 1999**

[54] **SECURE FIREWALL SUPPORTING DIFFERENT LEVELS OF AUTHENTICATION BASED ON ADDRESS OR ENCRYPTION STATUS**

[75] Inventors: **Spence Minear**, Fridley; **Edward B. Stockwell**, St. Paul; **Troy de Jongh**, Bloomington, all of Minn.

[73] Assignee: **Secure Computing Corporation**, Roseville, Minn.

[*] Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

[21] Appl. No.: **08/715,343**

[22] Filed: **Sep. 18, 1996**

[51] **Int. Cl.^{6}** **H04K 1/00**

[52] **U.S. Cl.** **713/201**; 380/49; 709/225; 709/249

[58] **Field of Search** 380/25, 42, 49; 395/187.01, 200.55, 200.6, 200.76, 200.79; 713/201; 709/225, 249

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,956,615	5/1976	Anderson et al.	380/24
4,104,721	8/1978	Markstein et al.	711/164
4,177,510	12/1979	Appell et al.	711/163
4,442,484	4/1984	Childs, Jr. et al.	711/163
4,584,639	4/1986	Hardy .	

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0 554 182 A1	4/1993	European Pat. Off. .
0 743 777 A2	11/1996	European Pat. Off. .
97/26734	7/1994	WIPO .
96/13113	5/1996	WIPO .
96/31035	10/1996	WIPO .
97/13340	4/1997	WIPO .
97/16911	5/1997	WIPO .
97/23972	7/1997	WIPO .
97/26731	7/1997	WIPO .
97/26735	7/1997	WIPO .
97/29413	8/1997	WIPO .

OTHER PUBLICATIONS

White, L.J., et al., "A Firewall Concept for Both Control–Flow and Data–Flow in Regression Integration Testing", *IEEE,* 262–271 (1992).

Merenbloom, P., "Network 'Fire Walls' Safeguard LAN Data from Outside Intrusion", *Infoworld,* p. 69 (Jul. 25, 1994).

Metzger, P., et al., "IP Authentication using Keyed MD5", RFC 1828, Piermont Information Services, Inc., New York, NY, http//ds.internic.net/rfc/rfc1828.txt, 6 p. (Aug. 1995).

Obraczka, K., et al., "Internet Resource Discovery Services", *Computer,* 26, 8–22 (Sep. 1993).

Press, L., "The Net: Progress and Opportunity", *Communications of the ACM,* 35, 21–25 (Dec. 1992).

Schroeder, M.D., et al., "A Hardware Architecture for Implementing Protection Rings", *Communications of the ACM,* 15, 157–170 (Mar. 1972).

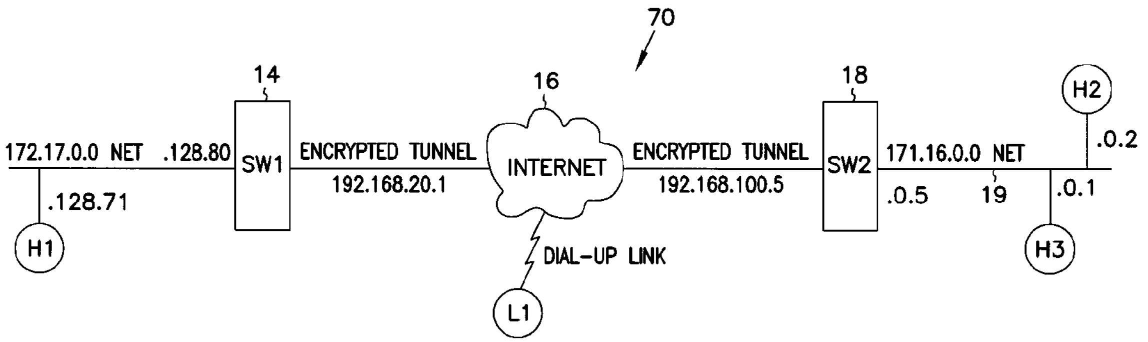
(List continued on next page.)

Primary Examiner—Kenneth S. Kim
Attorney, Agent, or Firm—Schwegman, Lundberg, Woessner & Kluth, P.A.

[57] **ABSTRACT**

A system and method for regulating the flow of messages through a firewall having a network protocol stack, wherein the network protocol stack includes an Internet Protocol (IP) layer, the method comprising establishing a security policy, determining, at the IP layer, if a message is encrypted, if the message is not encrypted, passing the unencrypted message up the network protocol stack to an application level proxy, and if the message is encrypted, decrypting the message and passing the decrypted message up the network protocol stack to the application level proxy, wherein decrypting the message includes executing a process at the IP layer to decrypt the message.

16 Claims, 5 Drawing Sheets



U.S. PATENT DOCUMENTS

4,621,321	11/1986	Boebert et al.	707/8
4,648,031	3/1987	Jenner et al. .	
4,701,840	10/1987	Boebert et al. .	
4,713,753	12/1987	Boebert et al.	711/164
4,870,571	9/1989	Frink	709/224
4,885,789	12/1989	Burger et al.	382/25
5,093,914	3/1992	Coplien et al.	395/704
5,124,984	6/1992	Engel	370/230
5,153,918	10/1992	Tuai	380/25
5,204,961	4/1993	Barlow	380/4
5,228,083	7/1993	Lozowick et al.	380/9
5,263,147	11/1993	Francisco et al.	711/164
5,272,754	12/1993	Boebert	380/25
5,276,735	1/1994	Boebert et al.	380/21
5,303,303	4/1994	White	380/49
5,305,385	4/1994	Schanning ct al.	380/49
5,311,593	5/1994	Carmi	380/23
5,329,623	7/1994	Smith et al.	711/214
5,333,266	7/1994	Boaz et al.	709/206
5,355,474	10/1994	Thuraisngham et al.	707/9
5,414,833	5/1995	Hershey et al.	380/4
5,416,842	5/1995	Aziz	380/30
5,485,460	1/1996	Schrier et al.	709/227
5,511,122	4/1996	Atkinson	380/25
5,530,758	6/1996	Marino, Jr. et al.	380/49
5,548,646	8/1996	Aziz et al.	380/23
5,550,984	8/1996	Gelb	709/245
5,566,170	10/1996	Bakke et al.	370/392
5,583,940	12/1996	Vidrascu et al.	380/49
5,586,260	12/1996	Hu	395/500
5,604,490	2/1997	Blakley, III et al.	340/825.31
5,606,668	2/1997	Shwed	380/42
5,615,340	3/1997	Dai et al.	709/250
5,619,648	4/1997	Canale et al.	709/206
5,623,601	4/1997	Vu	713/202
5,636,371	6/1997	Yu	395/500
5,644,571	7/1997	Seaman	370/401
5,671,279	9/1997	Elgalmal et al.	380/23
5,673,322	9/1997	Pepe et al.	380/49
5,684,951	11/1997	Goldman et al.	707/9
5,689,566	11/1997	Nguyen	380/25
5,699,513	12/1997	Feigen et al.	713/200
5,706,507	1/1998	Schloss	707/104
5,720,035	2/1998	Allegre et al.	709/225
5,781,550	7/1998	Templin et al.	370/401

OTHER PUBLICATIONS

Schwartz, M.F., "Internet Resource Discovery at the University of Colorado", *Computer,* 26, 25–35 (Sep. 1993).

Smith, R.E., "Sidewinder: Defense in Depth Using Type Enforcement", *International Journal of Network Management,* 219–229, (Jul.–Aug. 1995).

Thomsen, D., "Type Enforcement: The New Security Model", *Proceedings of the SPIE, Multimedia: Full–Service Impact on Business, Education and the Home,* vol. 2617, Philadelphia, PA, 143–150 (Oct. 23–24, 1995).

Warrier, U.S., et al., "A Platform for Heterogeneous Interconnection Network Management", *IEEE Journal on Selected Areas in Communications,* 8, 119–126 (Jan. 1990).

Wolfe, A, "Honeywell Builds Hardware for Computer Security", *Electronics,* 14–15 (Sep. 2, 1985).

Damashek, M., "Gauging Similarity with n–Grams: Language–Independent Categorization of Text", *Science,* 267, 843–848 (Feb. 10, 1995).

Dillaway, B.B., et al., "A Practical Design For A Multilevel Secure Database Management System", American Institute of Aeronautics and Astronautics, Inc., 44–57 (Dec. 1986).

Fine, T., et al., "Assuring Distributed Trusted Mach", *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy,* 206–218 (1993).

Foltz, P.W., et al., "Personalized Information Delivery: An Analysis of Information Filtering Methods", *Communication of the ACM,* 35, 51–60 (Dec. 1992).

Goldberg, D., et al., "Using Collaborative Filtering to Weave an Information Tapestry", *Communications of the ACM,* 35, 61–70 (Dec. 1992).

Grampp, F.T., "UNIX Operating System Security", *AT&T Bell Laboratories Technical Journal,* 63, 1649–1672 (Oct. 1984).

Haigh, J.T., et al., "Extending the Non–Interference Version of MLS for SAT", Proceedings of the 1996 IEEE Symposium on Security and Privacy, Oakland, CA, 232–239 (Apr. 7–9, 1986).

Karn, P., et al., "The ESP DES–CBC Transform", RFC 1829, Qualcomm, Inc., San Diego, CA, http//ds.internic.net/rfc/rfc1829.txt, 11 p. (Aug. 1995).

Kent, S.T., "Internet Privacy Enhanced Mail", Communications of the ACM, 36, 48–60 (Apr. 1993).

Lampson, B.W., "Dynamic Protection Structures", AFIPS Conference Proceedings, vol. 35, 1969 Fall Joint Computer Conference, Las Vegas, NV, 27–38 (Nov. 18–20, 1969).

Lee, K.–C., et al., "A Framework for Controlling Cooperative Agents", *Computer,* 8–16 (Jul. 1993).

Locb, S., "Architccting Pcrsonalizcd Dclivcry of Multimcdia Information", *Communications of the ACM,* 35, 39–50 (Dec. 1992).

Locb, S., ct al., "Information Filtcring," *Communications of the ACM,* 35, 26–28 (Dec. 1992).

PCT Search Report, Application No. PCT/US 95/12681, 8 p. (Apr. 4, 1996).

"Sidewinder Internals", Product Information, Secure Computing Corporation, 16 p. (Oct. 12, 1994).

"Special Report: Secure Computing Corporation and Network Security", *Computer Select,* 13 p. (Dec. 1995).

Cobb, S., "Establishing Firewall Policy", *IEEE,* 198–205 (1996).

Gassman, B., "Internet Security, and Firewalls Protection on the Internet", *IEEE,* 93–107 (1996).

Greenwald, M., et al., "Designing an Academic Firewall: Policy, Practice, and Experience with SURF", *IEEE,* 79–92 (1996).

McCarthy, S.P., "Hey Hackers! Secure Computing Says You Can't Break into This Telnet Site", *Computer Select,* 2 p. (Dec. 1995).

Peterson, L.L., et al., Computer Networks: A Systems Approach, Morgan Kaufmann Publishers, Inc., San Francisco, CA, pp. 218–221, 284–286 (1996).

Smith, R.E., "Constructing a High Assurance Mail Guard", Secure Computing Corporation (Appeared in the Proceedings of the National Computer Security Conference), 7 p. (1994).

Stempel, S., "IpAccess—An Internet Service Access System for Firewall Installations", *IEEE,* 31–41 (1995).

"SATAN No Threat to Sidewinder™", News Release, Secure Computing Corporation (Apr. 26, 1995).

"Answers to Frequently Asked Questions About Network Security", Secure Computing Corporation, 41 p. (1994).

Adam, J.A., "Meta–matrices", *IEEE Spectrum,* 26–27 (Oct. 1992).

Adam, J.A., "Playing on the Net", IEEE Spectrum, 29 (Oct. 1992).

Ancilotti, P., et al., "Language Features for Access Control", *IEEE Transactions on Software Engineering,* SE–9, 16–25 (Jan. 1983).

Atkinson, R., "IP Authentication Header", RFC 1826, Naval Research Laboratory, Washington, D.C., http/ds.internic.net/rfc/rfc1826.txt, 13 p. (Aug. 1995).

Atkinson, R., "IP Encapsulating Security Payload (ESP)", RFC 1827, Naval Research Laboratory, Washington, D.C., http//ds.internic.net/rfc/rfc1827.txt, 12 p. (Aug. 1995).

Atkinson, R., "Security Architecture for the Internet Protocol", RFC 1825, Naval Research Laboratory, Washington, D.C., http//ds.internic.net/rfc/rfc1825.txt, 21 p. (Aug. 1995).

Badger, L., et al., "Practical Domain and Type Enforcement for UNIX", Proceedings of the 1995 IEEE Symposium on Security and Privacy, Oakland, CA 66–77 (May 8–10, 1995).

Belkin, N.J., et al., "Information Filtering and Information Retrieval: Two Sides of the Same Coin?", *Communications of the ACM,* 35, 29–38 (Dec. 1992).

Bellovin, S.M., et al., "Network Firewalls", *IEEE Communications Magazine,* 32, 50–57 (Sep. 1994).

Bevier, W.R., et al., "Connection Policies and Controlled Interference", Proceedings of the 8th IEEE Computer Security Foundations Workshop, Kenmare, County Kerry, Ireland, 167–176 (Jun. 13–15, 1995).

Bowen, T.F., et al., "The Datacycle Architecture", *Communications of the ACM,* 35, 71–81 (Dec. 1992).

Bryan, J., "Firewalls For Sale", *BYTE,* pp. 99–100, 102 and 104 (Apr. 1995).

"100% of Hackers Failed to Break Into One Internet Site Protected by Sidewinder", News Release, Secure Computing Corporation (Feb. 16, 1995).

"Internet Security System Given 'Product of the Year' Award", News Release, Secure Computing Corporation, (Mar. 28, 1995).

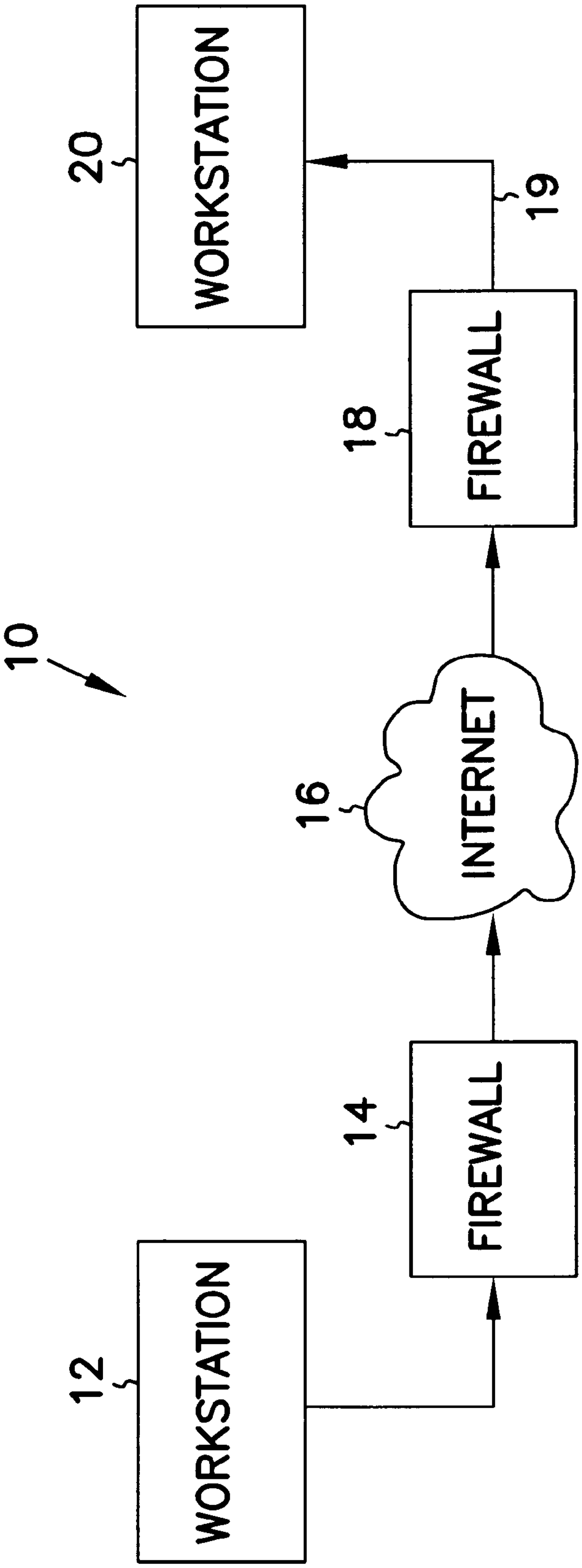

10
12
WORKSTATION
14
FIREWALL
16
INTERNET
18
FIREWALL
19
20
WORKSTATION

FIG. 1

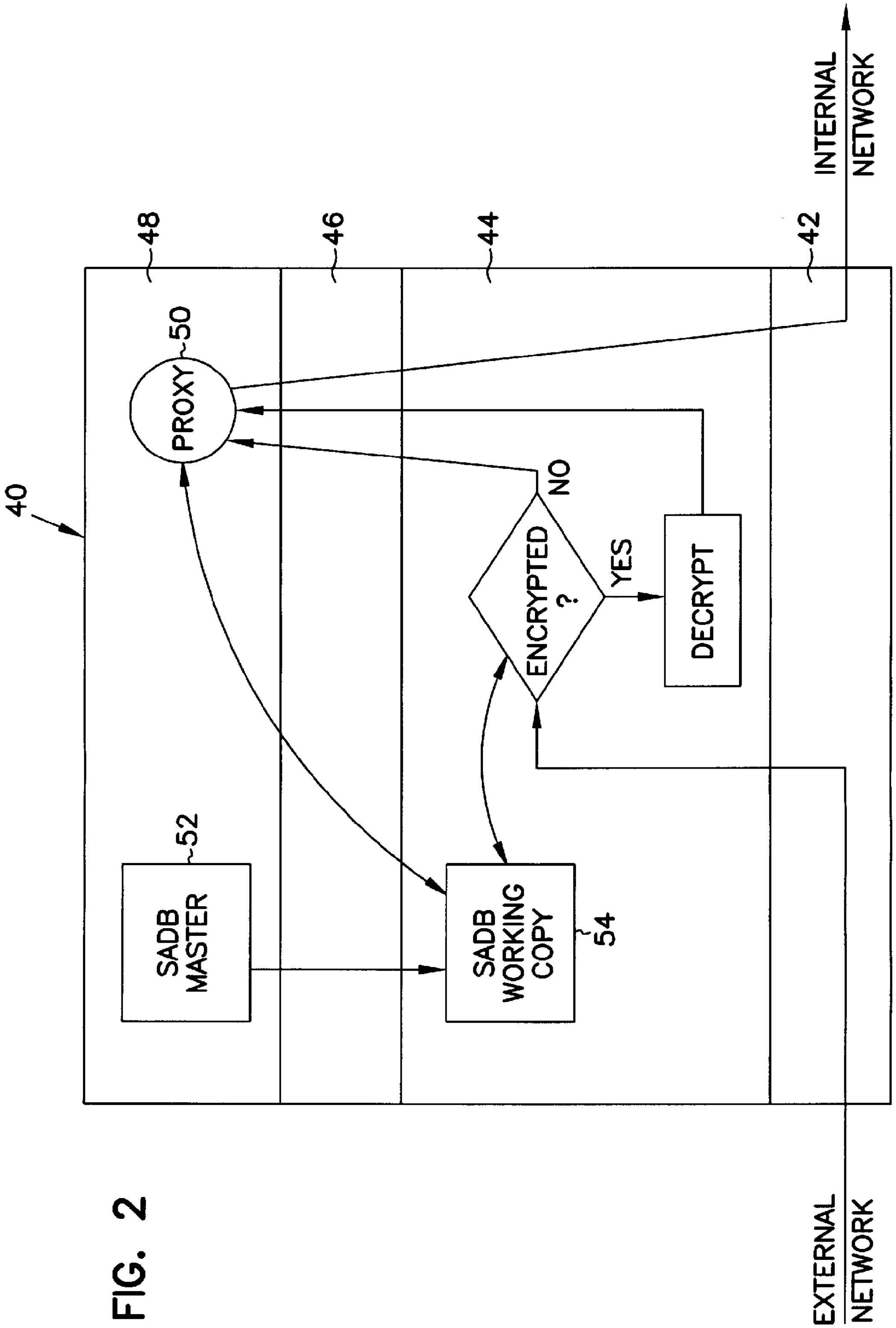

40
48
46
44
42
PROXY
50
NO
ENCRYPTED ?
YES
DECRYPT
SADB MASTER
52
SADB WORKING COPY
54
INTERNAL NETWORK
EXTERNAL NETWORK

FIG. 2

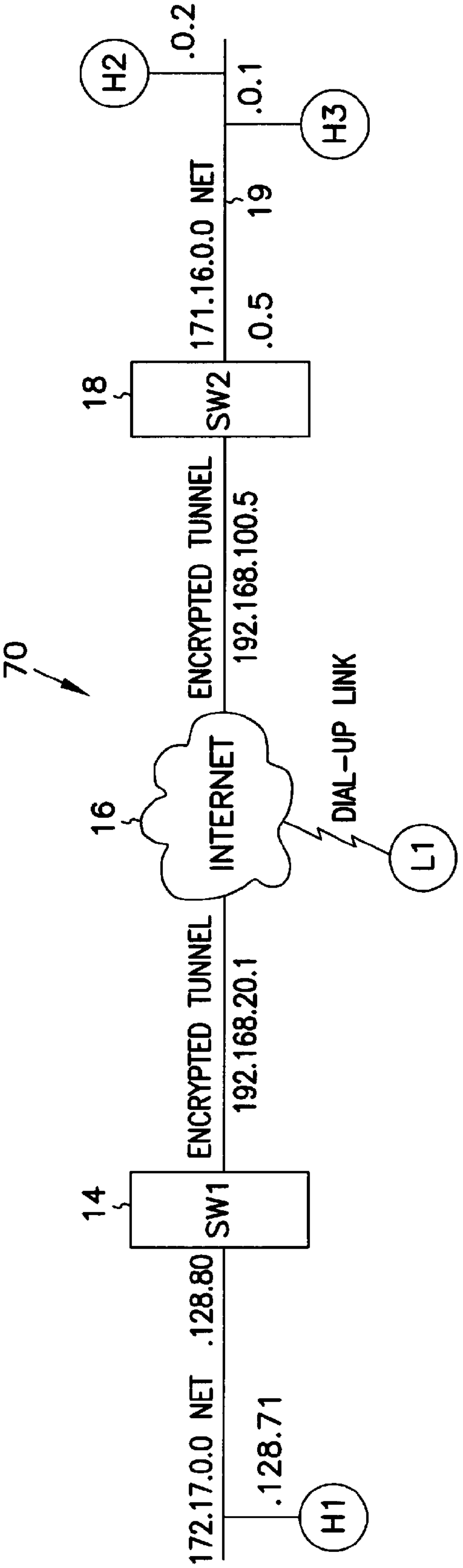

70
H1
.128.71
172.17.0.0 NET
.128.80
14
SW1
ENCRYPTED TUNNEL
192.168.20.1
16
INTERNET
DIAL-UP LINK
L1
ENCRYPTED TUNNEL
192.168.100.5
18
SW2
171.16.0.0 NET
.0.5
19
.0.1
H3
.0.2
H2

FIG. 3

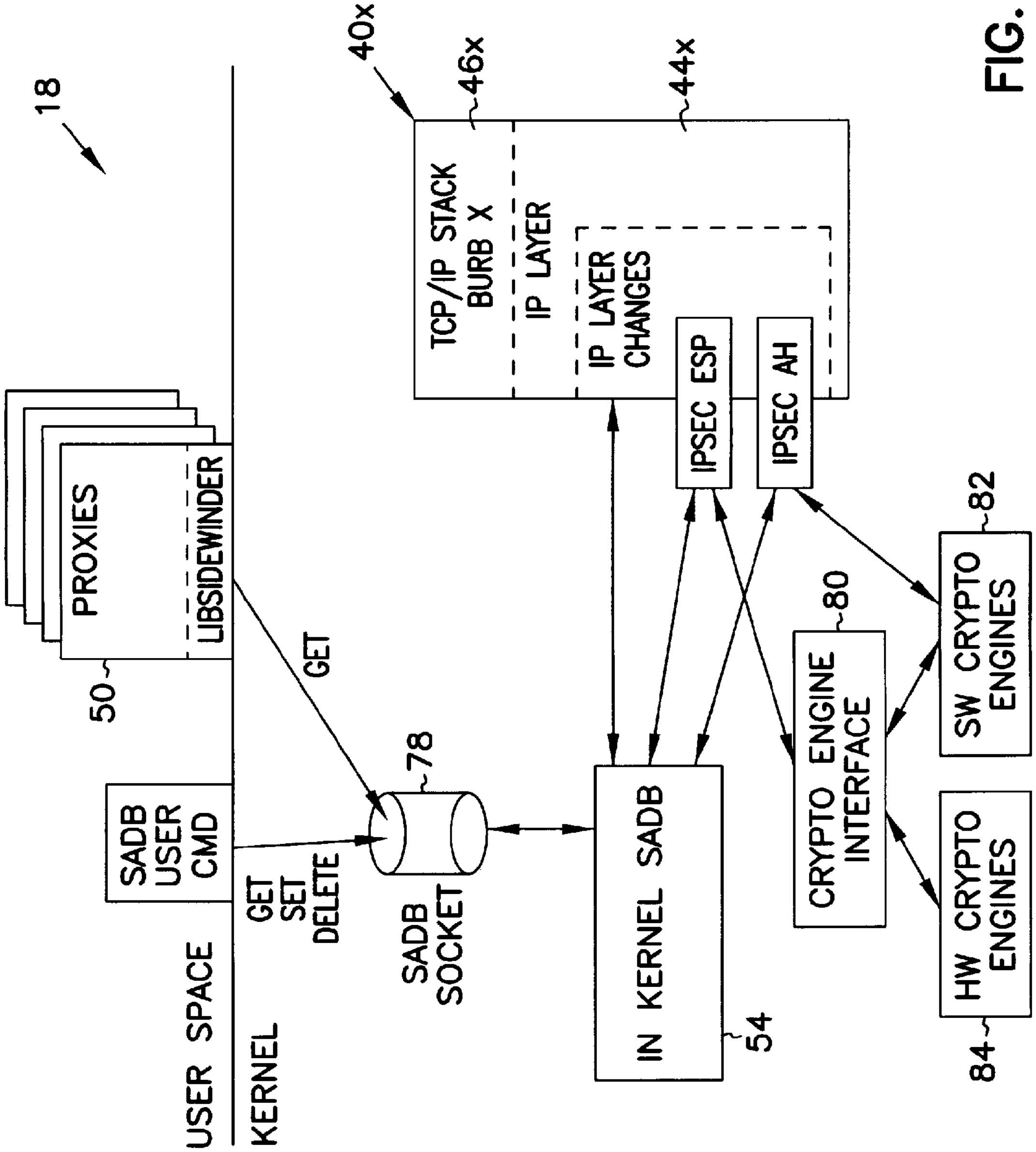

18
USER SPACE
KERNEL
PROXIES
LIBSIDEWINDER
50
SADB USER CMD
GET
SET
DELETE
GET
SADB SOCKET
78
IN KERNEL SADB
54
40x
46x
44x
TCP/IP STACK BURB X
IP LAYER
IP LAYER CHANGES
IPSEC ESP
IPSEC AH
CRYPTO ENGINE INTERFACE
80
SW CRYPTO ENGINES
82
HW CRYPTO ENGINES
84

FIG. 4

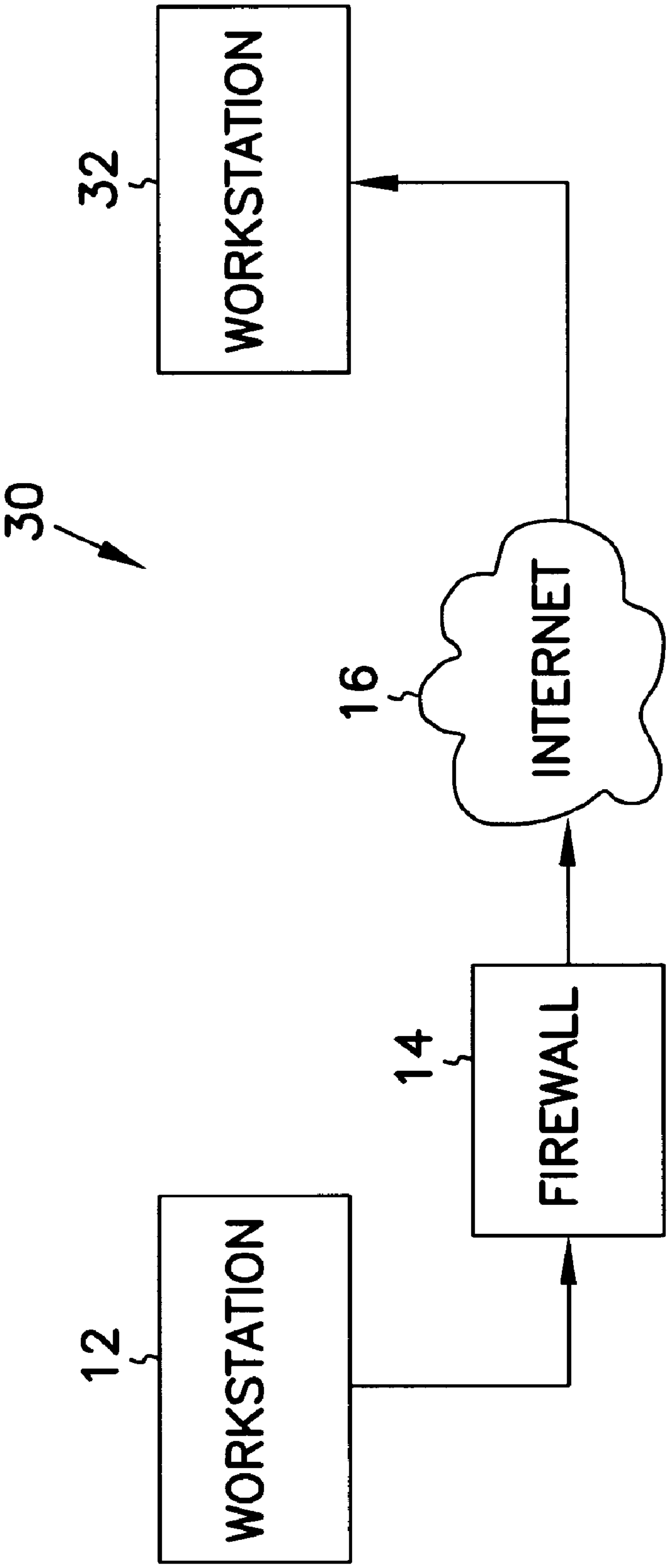


FIG. 5

SECURE FIREWALL SUPPORTING DIFFERENT LEVELS OF AUTHENTICATION BASED ON ADDRESS OR ENCRYPTION STATUS

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention pertains generally to network communications, and in particular to a system and method for securely transferring information between firewalls over an unprotected network.

2. Background Information

Firewalls have become an increasingly important part of network design. Firewalls provide protection of valuable resources on a private network while allowing communication and access with systems located on an unprotected network such as the Internet. In addition, they operate to block attacks on a private network arriving from the unprotected network by providing a single connection with limited services. A well designed firewall limits the security problems of an Internet connection to a single firewall computer system. This allows an organization to focus their network security efforts on the definition of the security policy enforced by the firewall. An example of a firewall is given in "SYSTEM AND METHOD FOR PROVIDING SECURE INTERNETWORK SERVICES", U.S. patent application Ser. No. 08/322078, filed Oct. 12, 1994 by Boebert et al., now issued U.S. Pat. No. 5,864,683, the description of which is hereby incorporated by reference. A second example of such a system is described in "SYSTEM AND METHOD FOR ACHIEVING NETWORK SEPARATION", U.S. application Ser. No. 08/599,232, filed Feb. 9, 1996 by Gooderum et al., the description of which is hereby incorporated by reference. Both are examples of application level gateways. Application level gateways use proxies operating at the application layer to process traffic through the firewall. As such, they can review not only the message traffic but also message content. In addition, they provide authentication and identification services, access control and auditing.

Data to be transferred on unprotected networks like the Internet is susceptible to electronic eavesdropping and accidental (or deliberate) corruption. Although a firewall can protect data within a private network from attacks launched from the unprotected network, even that data is vulnerable to both eavesdropping and corruption when transferred from the private network to an external machine. To address this danger, the Internet Engineering Task Force (IETF) developed a standard for protecting data transferred between firewalls over an unprotected network. The Internet Protocol Security (IPSEC) standard calls for encrypting data before it leaves the first firewall, and then decrypting the data when it is received by the second firewall. The decrypted data is then delivered to its destination, usually a user workstation connected to the second firewall. For this reason IPSEC encryption is sometimes called firewall-to-firewall encryption (FFE) and the connection between a workstation connected to the first firewall and a client or server connected to the second firewall is termed a virtual private network, or VPN.

The two main components of IPSEC security are data encryption and sender authentication. Data encryption increases the cost and time required for the eavesdropping party to read the transmitted data. Sender authentication ensures that the destination system can verify whether or not the encrypted data was actually sent from the workstation that it was supposed to be sent from. The IPSEC standard defines an encapsulated payload (ESP) as the mechanism used to transfer encrypted data. The standard defines an authentication header (AH) as the mechanism for establishing the sending workstation's identity.

Through the proper use of encryption, the problems of eavesdropping and corruption can be avoided; in effect, a protected connection is established from the internal network connected to one firewall through to an internal network connected to the second firewall. In addition, IPSEC can be used to provide a protected connection to an external computing system such as a portable personal computer.

IPSEC encryption and decryption work within the IP layer of the network protocol stack. This means that all communication between two IP addresses will be protected because all interfirewall communication must go through the IP layer. Such an approach is preferable over encryption and decryption at higher levels in the network protocol stack since when encryption is performed at layers higher than the IP layer more work is required to ensure that all supported communication is properly protected. In addition, since IPSEC encryption is handled below the Transport layer, IPSEC can encrypt data sent by any application. IPSEC therefore becomes a transparent add-on to such protocols as TCP and UDP.

Since, however, IPSEC decryption occurs at the IP layer, it can be difficult to port IPSEC to an application level gateway while still maintaining control at the proxy over authentication, message content, access control and auditing. Although the IPSEC specification in RFC 1825 suggests the use of a mandatory access control mechanism in a multi-level secure (MLS) network to compare a security level associated with the message with the security level of the receiving process, such an approach provides only limited utility in an application level gateway environment. In fact, implementations on application level gateways to date have simply relied on the fact that the message was IPSEC-encrypted as assurance that the message is legitimate and have simply decoded and forwarded the message to its destination. This creates, however, a potential chink in the firewall by assuming that the encrypted communication has access to all services.

What is needed is a method of handling IPSEC messages within an application level gateway which overcomes the above deficiencies. The method should allow control over access by an IPSEC connection to individual services within the internal network..

SUMMARY OF THE INVENTION

The present invention is a system and method for regulating the flow of messages through a firewall having a network protocol stack, wherein the network protocol stack includes an Internet Protocol (IP) layer, the method comprising the steps of determining, at the IP layer, if a message is encrypted, if the message is not encrypted, passing the unencrypted message up the network protocol stack to an application level proxy, and if the message is encrypted, decrypting the message and passing the decrypted message up the network protocol stack to the application level proxy, wherein the step of decrypting the message includes the step of executing a procedure at the IP layer to decrypt the message.

According to another aspect of the present invention, a system and method is described for authenticating the sender of a message within a computer system having a network protocol stack, wherein the network protocol stack includes an Internet Protocol (UP) layer, the method comprising the steps of determining, at the IP layer, if the message is encrypted, if the message is encrypted, decrypting the message, wherein the step of decrypting the message includes the step of executing a procedure at the IP layer to decrypt the message, passing the decrypted message up the network protocol stack to an application level proxy, determining an authentication protocol appropriate for the message, and executing the authentication protocol to authenticate the sender of the message.

BRIEF DESCRIPTION OF THE DRAWINGS

In the drawings, where like numerals refer to like components throughout the several views:

FIG. **1** is a functional block diagram of an application level gateway-implemented firewall-to-firewall encryption scheme according to the present invention;

FIG. **2** is a block diagram showing access control checking of both encrypted and unencrypted messages in network protocol stack according to the present invention;

FIG. **3** is a block diagram of a representative application level gateway-implemented firewall-to-firewall encryption scheme;

FIG. **4** is a block diagram of one embodiment of a network-separated protocol stack implementing IPSEC according to the present invention; and

FIG. **5** is a functional block diagram of a firewall-to-workstation encryption scheme according to the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the following detailed description of the preferred embodiment, references made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific preferred embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention, and it is to be understood that other embodiments may be utilized and that structural, logical, physical, architectural, and electrical changes may be made without departing from the spirit and scope of the present invention. The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the present invention is defined only by the appended claims and their equivalents.

A system **10** which can be used for firewall-to-firewall encryption (FFE) is shown in FIG. **1**. In FIG. **1**, system **10** includes a workstation **12** communicating through a firewall **14** to an unprotected network **16** such as the Internet. System **10** also includes a workstation **20** communicating through a firewall **18** to unprotected network **16**. In one embodiment, firewall **18** is an application level gateway.

As noted above, IPSEC encryption and decryption work within the IP layer of the network protocol stack. This means that all communications between two IP addresses will be protected because all interfirewall communication must pass through the IP layer. IPSEC takes the standard Internet packet and converts it into a carrier packet. The carrier packet is designed to do two things: to conceal the contents of the original packet (encryption) and to provide a mechanism by which the receiving firewall can verify the source of the packet (authentication). In one embodiment of the present invention, each IPSEC carrier packet includes both an authentication header used to authenticate the sending machine and an encapsulated payload containing encrypted data. The authentication header and the encapsulated payload features of IPSEC can, however, be used independently. As required in RFC 1825, DES-CBC is provided for use in encrypting the encapsulated payload while the authentication header uses keyed MD5.

To use IPSEC, you must create a security association (SA) for each destination IP address. In one embodiment, each SA contains the following information:

Security Parameters Index (SPI)—The index used to find a SA on receipt of an IPSEC datagram.

Destination IP address—The address used to find the SA and trigger use of IPSEC processing on output.

The peer SPI—The SPI value to put on a IPSEC datagram on output.

The peer IP address—The destination IP address to be put into the packet header if IPSEC Tunnel mode is used.

The Encryption Security Payload (ESP) algorithm to be used.

The ESP key to used for decryption of input datagrams.

The ESP key to used for encryption of output datagrams.

The authentication (AH) algorithm to be used.

The AH key to be used for validation of input packets.

The AH key to be used for generation of the authentication data for output datagrams.

The combination of a given Security Parameter Index and Destination IP address uniquely identifies a particular "Security Association." In one embodiment, the sending firewall uses the sending userid and Destination Address to select an appropriate Security Association (and hence SPI value). The receiving firewall uses the combination of SPI value and Source address to obtain the appropriate Security Association.

A security association is normally one-way. An authenticated communications session between two firewalls will normally have two Security Parameter Indexes in use (one in each direction). The combination of a particular Security Parameter Index and a particular Destination Address uniquely identifies the Security Association.

More information on the specifics of an IPSEC FFE implementation can be obtained from the standards developed by the IPSEC work group and documented in *Security Architecture for IP* (RFC 1825) and in RFC's 1826–1829.

When a datagram is received from unprotected network **16** or is to be transmitted to a destination across unprotected network **16**, the firewall must be able to determine the algorithms, keys, etc. that must be used to process the datagram correctly. In one embodiment, this information is obtained via a security association lookup. In one such embodiment, the lookup routine is passed several arguments: the source IP address if the datagram is being received from network **16** or the destination IP address if the datagram is to be transmitted across network **16**, the SPI, and a flag that is used to indicate whether the lookup is being done to receive or transmit a datagram.

When an IPSEC datagram is received by firewall **18** from unprotected network **16**, the SPI and source IP address are determined by looking in the datagram. In one embodiment a Security Association Database (SADB) stored within firewall **18** is searched for the entry with a matching SPI. In one such embodiment, security associations can be set up based on network address as well as a more granular host address. This allows the network administrator to create a security association between two firewalls with only a couple of lines in a configuration file on each machine. For such embodiments, the entry in the Security Association Database that has both the matching SPI and the longest address match is selected as the SA entry. In another such embodiment, each SA has a prefix length value associated with the address. An address match on a SA entry means that the addresses match for the number of bits specified by the prefix length value.

There are two exceptions to this search process. First, when an SA entry is set marked as being dynamic it implies that the user of this SA may not have a fixed IP address. In this case the match is fully determined by the SPI value. Thus it is necessary that the SPI values for such SA entries be unique in the SADB. The second exception is for SA entries marked as tunnel mode entries. In this case it is normally the case that the sending entity will hide its source address so that all that is visible on the public wire is the destination address. In this case, like in the case where the SA entries are for dynamic IP addresses, the search is done exclusively on the basis of the SPI.

When transmitting a datagram across unprotected network **16** the SADB is searched using only the destination address as an input. In this case the entry which has the longest address match is selected and returned to the calling routine.

In one embodiment, if firewall **18** receives datagrams which are identified as either an IP_PROTO_IPSEC_ESP or IP_PROTO_IPSEC_AH protocol datagram, there must be a corresponding SA in the SADB or else firewall **18** will drop the packet and an audit message will be generated. Such an occurrence might indicate a possible attack or it might simply be a symptom of an erroneous key entry in the Security Association Database.

In a system such as system **10**, application level gateway firewall **18** acts as a buffer between unprotected network **16** and workstations such as workstation **20**. Messages coming from unprotected network **16** are reviewed and a determination is made as to whether execution of an authentication and identification protocol is warranted. In contrast to previous systems, system **10** also performs this same determination on IPSEC-encrypted messages. If desired, the same authentication and identification can be made on messages to be transferred from workstation **20** to unprotected network **16**. FIG. **2** illustrates one way of authenticating both encrypted and unencrypted messages in a system such as system **10**.

In the system of FIG. **2** a network protocol stack 40 includes a physical layer 42, an Internet protocol (IP) layer 44, a Transport layer 46 and an application layer 48. Such a protocol stack exists, for instance on application level gateway firewall 18 of FIG. 1. An application executing in application layer 48 can communicate to an application executing on another system by preparing a message and transmitting it through one of the existing transport services executing on transport layer 46. Transport layer 46 in turn uses a process executing in IP layer 44 to continue the transfer. Physical layer 42 provides the software needed to transfer data through the communication hardware (e.g., a network interface card or a modem). As noted above, IPSEC executes within IP layer 44. Encryption and authentication is transparent to the host as long as the network administrator has the Security Association Database correctly configured and a key management mechanism is in place on the firewall.

In application level gateway firewall **18**, a proxy **50** operating within application layer **48** processes messages transferred between internal and external networks. All network-to-network traffic must pass through one of the proxies within application layer **48** before being the transfer across networks is allowed. A message arriving from external network **16** is examined at IP layer **44** and an SADB is queried to determine if the source address and SPI are associated with an SA. In the embodiment shown in FIG. **2**, an SADB Master copy **52** is maintained in persistent memory at application layer **48** while a copy **54** of SADB is maintained in volatile memory within the kernel. If the message is supposed to be encrypted, the message is decrypted based on the algorithm and key associated with the particular SA and the message is transferred up through transport layer **46** to proxy **50**. Proxy **50** examines the source and destination addresses and the type of service desired and decides whether authentication of the sender is warranted. If so, proxy **50** initiates an authentication protocol. The protocol may be as simple as requesting a user name and password or it may include a challenge/response authentication process. Proxy **50** also looks to see whether the message coming in was encrypted or not and may factor that into whether a particular type of authentication is needed. In Telnet, for instance, user name/password authentication may be sufficient for an FFE link while the security policy may dictate that a more stringent challenge/response protocol is needed for unencrypted links. In that case, proxy **50** will be a Telnet proxy and it will base its authentication protocol on whether the link was encrypted or not.

Since IPSEC executes within IP layer **44** there is no need for host firewalls to update their applications. Users that already have IPSEC available on their own host machine will, however, have to request that the firewall administrator set up SA's in the SADB for their traffic.

In the embodiment shown in FIG. **2**, a working copy **54** of the Security Association Database consisting of all currently active SA's is kept resident in memory for ready access by IP layer processing as datagrams are received and transmitted. In addition, a working master copy **52** of the SADB is maintained in a file in nonvolatile memory. During system startup and initialization processing the content of all of the required SA's in master SADB **52** is added to the working copy **54** stored in kernel memory.

In one embodiment, firewall **18** maintains different levels of security on internal and external network interfaces. It is desirable for a firewall to have different levels of security on both the internal and external interfaces. In one embodiment, firewall **18** supports three different levels, numbered 0 through 2. These levels provide a simple policy mechanism that controls permission for both in-bound and out-bound packets.

Level 0—do not allow any in-bound or out-bound traffic unless there is a security association between the source and destination.

Level 1—Allow both in-bound and out-bound non-IPSEC traffic but force the use of IPSEC if a SA exists for the address. (To support this firewall **18** must look for a SA for each in-bound datagram.)

Level 2—allow NULL security associations to exist. NULL associations are just like normal security associations, except no encryption or authentication transform is performed on in-bound or out-bound packets that correspond to this NULL association. With Level 2 enabled, the machine will still receive unprotected traffic, but it will not transmit unless Level 1 is enabled.

The default protection level established when the Security Association Database (SADB) is initialized at boot time is 1 for in-bound traffic and 2 for out-bound traffic.

It is desirable for user-level daemons or proxies in firewall **18** to know whether or not a connection (or incoming UDP datagram in the case of a UDP proxy) is being encrypted or not. This information may be used, for example, to control access to services within internal network **19** or to determine the authentication method that will be used to authenticate the sender. One such method of controlling access to services within an internal network is described in GENERALIZED SECURITY POLICY MANAGEMENT SYSTEM AND METHOD, U.S. patent application Ser. No. 08/715,668, filed herewith, which description is hereby incorporated by reference. In addition, the above-mentioned patent disclosure teaches a method of selecting an authentication method based on whether a message was encrypted or not. That description also is hereby incorporated by reference. In the case of a Level 0 policy, it would be safe to assume that all incoming traffic is encrypted or authenticated. In the case of Levels 1 through 2, a determination must be made whether or not a security association exists for a given peer. Otherwise an application may believe that in-bound traffic has been authenticated when it really has not. (That is why it is necessary to look for an SA on input of each non-IPSEC datagram.)

In one embodiment, a flag which accompanies the message as it is sent from IP layer **44** to proxy **50** indicates whether the incoming message was or was not encrypted. In another embodiment, proxy **50** accesses Security Association Database **54** (the table in the kernel can be queried via an SADB routing socket (PF-SADB)) to determine whether or not a security association exists for a given peer. The SADB socket is much like a routing socket found in the stock BSD 4.4 kernel (protocol family PF-ROUTE) except that PF-SADB sockets are used to maintain the Security Association Database (SADB) instead of the routing table. Because the private keys used for encryption, decryption, and keyed authentication are stored in this table, access must be strictly prohibited and allowed to only administrators and key management daemons. Care must be taken when allowing user-level daemons access to /dev/mem or /dev/kmem as well, since the keys are stored in kernel memory and could be exposed with some creative hacking.

In one embodiment, a command-line tool called sadb is used to support the generation and maintenance of in-kernel version **54** of SADB. The primary interface between this tool and the SADB is the PF-SADB socket. The kernel provides socket processing to receive client requests to add, update, or change entries in in-kernel SADB **54**. As noted above, the default protection level established when the Security Association Database (SADB) is initialized at boot time is 1 for in-bound traffic and 2 for out-bound traffic. This may be changed by the use of the sadb command.

The existing sadb command was derived from the NIST implementation of IPSEC. As noted above, this tool is much like route in that it uses a special socket to pass data structures in and out of the kernel. There are three commands recognized by the sadb command: get, set, delete. The following simple shell script supports adding and removing a single SA entry to SADB **54**. It shows one embodiment of a parameter order for adding a SA to the SADB.

```
# ! /bin/sh
if [ $# -ne 1 ]
then
    echo "usage: $0 <on>|<off>" >&2
    exit 1
fi
ONOFF=$1
addsa ()
{
IPADDRESS=$2
PEERADDRESS=0.0.0.0
PREFIXLEN=0                    # Num of bits, 0 => full 32 bit
match
LOCALADDRESS=0.0.0.0
REALADDRESS=0.0.0.0
PORT=0
PROTOCOL=0
UID=0
DESALG=1                       # I = DES-CBC
IVLEN=4                        # bytes
DESKEY=0b0b0b0b0b0b0b0b
DESKEYLEN=8                    # bytes
AHALG=1                        # 1 = MD5
AHKEY=30313233343536373031323334353637
AHKEYLEN=16                    # bytcs
LOCAL_SPI=$1
PEER_SPI=$1
TUNNEL_MODE=0
AHRESULTLEN=4
COMBINED_MODE=1                # On output, 1 = ESP, then AH;
0 = AH, then ESP
DYNAMIC_FLAG=0
if [ "$ONOFF" = "on"
then
    ./sadb add dst $IPADDRESS $PREFIXLEN $LOCAL_SPI
$UID $PEERADDRESS $PEER_SPI $TUNNEL_MODE
$LOCALADDRESS $REALADDRESS $PROTOCOL $PORT
$DESALG $IVLEN $DESKEYLEN $DESKEY $DESKEYLEN
$DESKEY $AHALG $AHKEYLEN $AHKEY $AHKEYLEN $AHKEY
$AHRESULTLEN $COMBINED_MODE $DYNAMIC_FLAG
else
    ./sadb delete dst $IPADDRESS $LOCAL-SPI
fi
}
#   Get down to work:
addsa 500 172.17.128.115          # number6.sctc.com
```

The current status of in-kernel SADB **54** can be obtained with the sadb command. The get option allows dumping the entire SADB or a single entry. In one embodiment, the complete dump approach uses /dev/kmem to find the information. The information may be presented as follows:

```
# sadb get dst
Local-SPI Address-Family Destination-Addr Preflx_length
UID
    Peer-Address Peer-SPI Transport-Type
    Local-Address Real-Address
    Protocol Port
    ESP_Alg_ID ESP_IVEC_Length
        ESP_Enc_Key_length ESP_Enc_ESP_Key
        ESP_Dec_Key_length ESP_Dec_ESP_Key
    AH_Alg_ID AH_Data_Length
        AH_Gen_Key_Length AH_Gen_Key
        AH_Check_Key_Length AH_Check_Key
    Combined_Mode Dynamic_Flag
---------------------------------------------------------------------------------
500 INET: number6.sctc.com 0 0
    0.0.0.0    500 Transport(0) 0
    0.0.0.0.0.0.0.0
    None None
    DES/CBC-RFC1829 (1) 4
        8 0b0b0b0b0b0b0b0b
        8 0b0b0b0b0b0b0b0b
    MD5-RFC1828 (1) 4
        16 30313233343536373031323334353637
        16 30313233343536373031323334353637
    ESP+AH(1) 0
501 INET: spokes.sctc.com 0 0
    0.0.0.0   501 Transport(0) 0
    0.0.0.0.0.0.0.0
    None None
    DES/CBC-RFC1829(1) 4
```

-continued

```
        8 0b0b0b0b0b0b0b0b
        8 0b0b0b0b0b0b0b0b
    MD5-RFC1828(1)4
        16 30313233343536373031323334353637
        16 30313233343536373031323334353637
    ESP+AH(1) 0
End of list.
```

When a new entry is added to in-kernel SADB **54**, the add process first checks to see that no existing entry will match the values provided in the new entry. If no match is found then the entry is added to the end of the existing SADB list.

To illustrate the use and administration of an FFE, we'll go through an example using FFE **70** in FIG. **3**. Firewalls **14** and **18** are both application level gateway firewalls implemented according to the present invention. Workstations H2 and H3 both want to communicate with H1. For the administrator of firewalls **14** and **18**, this is easy to accomplish. The administrator sets up a line something like this (we'll only show the IP address part and SPI parts of the SA, since they're the trickiest values to configure. Also, assume that we are using tunnel mode):

```
#   Hypothetical SW1 Config File
#
#   Fields are laid out in the following manner:
#   srcaddrornet= localSPI= peeraddr= peerSPI=
realsrcaddr= localaddr= key=
# The following entry sets up a tunnel between hosts
behind SW1
# and hosts behind SW2.
src=172.16.0.0 localSPI=666 peer=192.168.100.5
peerSPI=777 \
      realsrcaddr=192.168.100.5 localaddrs=0.0.0.0
key=0xdcadbccffadcbabc
#   IIypothetical SW2 config File
#
#   Fields are laid out in the following manner:
#   srcaddrornet= localSPI= peeraddr= peerSPI=
      realsrcaddr= localaddr= key=
#   The following entry sets up a tunnel between hosts
behind SW1 and
#   hosts behind SW2.
src=172.17.0.0 localSPI=777 peer=192.168.20.1
peerSPI=666 \
      realsrcaddr=192.168.20.1 localaddr=0.0.0.0 \
      key=Oxdeadbeeffadebabe
```

With this setup, all traffic is encrypted using one key, no matter who is talking to whom. For example, traffic from H2 to HI as well as traffic from H3 to H1 will be encrypted with one key. Although this setup is small and simple, it may not be enough.

What happens if H2 cannot trust H3? In this case, the administrator can set up security associations at the host level. In this case, we have to rely on the SPI field of the SA, since the receiving firewall cannot tell from the datagram header which host behind the sending firewall sent the packet. Since the SPI is stored in IPSEC datagrams, we can do a lookup to obtain its value. Below are the sample configuration files for both firewalls again, but this time, each host combination communicates with a different key. Moreover, H2 excludes H3 from communications with H1, and H3 excludes H2 in the same way.

```
#   Hypothetical SW1 Config File
#
#   Fields are laid out in the following manner:
#   srcaddrornet= localSPI= peeraddr= peerSPI=
realsrcaddr= localaddr= key=
# The following entry sets up a secure link between H2
and H1
src=172.16.0.2 localSPI=666 peer=192.168.100.5
peerSPI=777 \
      realsrcaddr=192.168.100.5 localaddrs=178.17.128.71
\
      key=0x0a0a0a0a0a0a0a0a
# The following entry sets up a secure link between H3
and H1
src=172.16.0.1 localSPI=555 peer=l92.l68.l00.5
peerSPI=888 \
      realsrcaddr=192.168.100.5 localaddrs=178.17.128.71
\
      key=0x0b0b0b0b0b0b0b0b
#   Hypothetical SW2 Config File
#
#   Fields are laid out in the following manner:
#   srcaddrornet= localSPI= peeraddr= peerSPI=
realsrcaddr= localaddr= key=
# The following entry sets up a secure link between H2
and H1
src=172.17.128.71 localSPI=777 peer=192.168.20.1
peerSPI=666 \
      realsrcaddr=192.168.20.1 localaddrs=172.16.0.2 \
      key=0x0a0a0a0a0a0a0a0a
# The following entry sets up a secure link between H3
and H1
src=172.17.128.71 localSPI=888 peer=192.168.20.1
peerSPI=555 \
      realsrcaddr=192.168.20.1 localaddrs=172.16.0.1 \
      key=0x0b0b0b0b0b0b0b0b
```

FIG. **4** is a block diagram showing in more detail one embodiment of an IPSEC-enabled application level gateway firewall **18**. Application level gateway firewall **18** provides access control checking of both encrypted and unencrypted messages in a more secure environment due to its network-separated architecture. Network separation divides a system into a set of independent regions or burbs, with a domain and a protocol stack assigned to each burb. Each protocol stack **40**x has its own independent set of data structures, including routing information and protocol information. A given socket will be bound to a single protocol stack at creation time and no data can pass between protocol stacks **40** without going through proxy space. A proxy **50** therefore acts as the go-between for transfers between domains. Because of this, a malicious attacker who gains control of one of the regions is prevented from being able to compromise processes executing in other regions. Network separation and its application to an application level gateway is described in "SYSTEM AND METHOD FOR ACHIEVING NETWORK SEPARATION", U.S. application Ser. No. 08/599,232, filed Feb. 9, 1996 by Gooderum et al., the description of which is hereby incorporated by reference.

In the system shown in FIG. **4**, the in-bound and out-bound datagram processing of a security association continues to follow the conventions defined by the network separation model. Thus all datagrams received on or sent to a given burb remain in that burb once decrypted. In one such embodiment SADB socket **78** has been defined to have the type 'sadb'. Each proxy **50** that requires access to SADB socket **78** to execute its query as to whether the received message was encrypted must have create permission to the sadb type.

The following is list of specific requirements that a system such as is shown in FIG. **4** must provide. Many of the requirements were discussed in the information provided earlier in this document.

1. Firewall applications may query the IPSEC subsystem to determine if traffic with a given address is guaranteed to be encrypted.

2. Receipt of an unencrypted datagram from an address that has a SA results in the datagram being dropped and an audit message being generated.
3. On receipt of encrypted protocol datagrams the SADB searches will be done using the SPI as the primary key. The source address will a secondary key. The SA returned by the search will be the SA which matches the SPI exactly and has the longest match with the address.
4. A search of the SADB for a SPI that finds an entry that is marked as SA for a dynamic IP will not consider the address in the search process.
5. A search of the SADB for a SPI that finds an entry that is marked as a SA for a tunnel mode connection will to consider the address if it is (0.0.0.0) i.e INADDR.
6. On receipt of a non-IPSEC datagram the SADB will be searched for an entry that matches the src address. If a SA is found the datagram will be dropped and an audit message sent.
7. SADB searches on output will be done using the DST address as key. If more than one SA entry in the SADB has that address the first one with the maximum address match will be returned.
8. The SADB must be structured so that searches are fast regardless if the search is done by SPI or by address.
9. The SADB must provide support for connections to a site with a fixed SPI but changing IP address. SA entries for such connections will be referred to as Dynamic Address Sites, or just Dynamic entries.
10. When a dynamic entry is found by a SPI search, the current datagram's SRC address, which is required to ensure that the return datagrams are properly encrypted, will be recorded in the SA only after the AH checking has passed successfully. (This is because if the address is recorded before AH passes then an attacker can cause return packets of an outgoing connection to be transmitted in the clear.)
11. A failure of an AH check on a dynamic entry results in an audit message.
12. In an embodiment where the firewall requires that all connections use both AH and ESP, on receipt the order should be AH first ESP second.
13. The processing structure on both input and output should try to minimize the number of SADB required lookups.

Returning to FIG. **4**, in one embodiment firewall **18** includes a crypto engine interface **80** used to encrypt an IPSEC payload. Crypto engine interface **80** may be connected to a software encryption engine **82** or to a hardware encryption engine **84**. Engines **82** and **84** perform the actual encryption function using, for example, DES-CBC. In addition, software encryption engine **82** may include the keyed MD5 algorithm used for AH.

In one embodiment, crypto engine interface **80** is a utility which provides a consistent interface between the software and hardware encryption engines. As shown in FIG. **4**, in one such embodiment interface **80** only supports the use of the use of hardware cryptographic engine **84** for IPSEC ESP processing. The significant design issue that interface **80** must deal with is that use of a hardware encryption engine requires that the processing be down in disjoint steps operating in different interrupt contexts as engine **84** completes the various processing steps.

The required information is stored in a request structure that is bound to the IP datagram being processed. The request is of type crypto_request_t. This structure is quite large and definitely does not contain a minimum state set.

In addition to the definition of the request data structure, this software implementing interface **80** provides two functions which isolate the decision of which cryptographic engine to use. The crypt_des_encrypt function is for use by the IP output processing to encrypt a datagram. The crypt_des_decrypt function is for use by the IP input processing to decrypt a datagram. If hardware encryption engine **84** is present and other hardware usage criteria are met the request is enqueued on a hardware processing queue and a return code indicating that the cryptographic processing is in progress is returned. If software engine **82** is used, the return code indicates that the cryptographic processing is complete. In the former case, the continuation of the IP processing is delayed until after hardware encryption is done. Otherwise it is completed as immediately in the same processing stream.

There are two software cryptographic engines **82** provided in the IPSEC software. One provides the MD**5** algorithm used by the IPSEC AH processing, and the other provides the DES algorithm used by the IPSEC ESP processing. This software can be obtained from the U.S. Government IPSEC implementation.

In one embodiment hardware cryptographic engine **84** is provided by a Cylink SafeNode processing board. The interface to this hardware card is provided by the Cylink device driver. A significant aspect of the Cylink card that plays a major part in the design of the IPSEC Cylink driver is that the card functions much like a low level subroutine interface and requires software support to initiate each processing step. Thus to encrypt or decrypt an individual datagram there are a minimum of two steps, one to set the DES initialization vector and one to do the encryption. Since the IP processing can not suspend itself and wait while the hardware completes and then be rescheduled by the hardware interrupt handler, in one embodiment a finite state machine is used to tie sequences of hardware processing elements together. In one such embodiment the interrupt handler looks at the current state, executes a defined after state function, transitions to the state and then executes that state's start function.

One function, cyl_enqueue_request, is used to initiate either an encrypt or a decrypt action. This function is designed to be called by cryptographic engine interface **80**. All of the information required to initiate the processing as well as the function to be performed after the encryption operation is completed is provided in the request structure. This function will enqueue the request on the hardware request queue and start the hardware processing if necessary.

A system **30** which can be used for firewall-to-workstation encryption is shown in FIG. **5**. In FIG. **5**, system **30** includes a workstation **12** communicating through a firewall **14** to an unprotected network **16** such as the Internet. System **30** also includes a workstation **32** communicating directly with firewall **14** through unprotected network **16**. Firewall **14** is an application level gateway incorporating IPSEC handling as described above. (It should be noted that IPSEC security cannot be used to authenticate the personal identity of the sender for a firewall to firewall transfer. When IPSEC is used, however, on a single user machine such as a portable personal computer, IPSEC usage should be protected with a personal identification number (PIN). In these cases IPSEC can be used to help with user identification to the firewall.)

According to the IPSEC RFC's, you can use either tunnel or transport mode with this embodiment based on your security needs. In certain situations, the communications must be sent in tunnel mode to hide unregistered addresses.

Although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that any arrangement which is calculated to achieve the same purpose may be substituted for the specific embodiment shown. This application is intended to cover any adaptations or variations of the present invention. Therefore, it is intended that this invention be limited only by the claims and the equivalents thereof.

What is claimed is:

1. A method of regulating the flow of messages between an external network and an internal network through a firewall having a network protocol stack, wherein the network protocol stack includes an Internet Protocol (IP) layer, the method comprising:

establishing a security policy;

determining, at the IP layer, if a message to an IP address is encrypted;

if the message to the IP address is not encrypted, passing the unencrypted message up the network protocol stack to an application level proxy;

if the message to the IP address is encrypted, decrypting the message and passing the decrypted message up the network protocol stack to the application level proxy, wherein decrypting the message includes executing a procedure at the IP layer to decrypt the message;

determining at the application level proxy and based on the security policy if the message to that IP address is one that can be forwarded, wherein the decision whether to forward is a function of whether the message was encrypted when received; and

passing the message from the application level proxy to its destination through the IP layer.

2. A method of authenticating the sender of a message within a computer system having a network protocol stack, wherein the network protocol stack includes an Internet Protocol (IP) layer, the method comprising:

providing a plurality of authentication protocols, wherein each authentication protocol provides a different level of security;

determining, at the IP layer, if the message is encrypted;

if the message is encrypted, decrypting the message, wherein decrypting the message includes executing a process at the IP layer to decrypt the message;

passing the decrypted message up the network protocol stack to an application level proxy;

selecting an authentication protocol from the plurality of authentication protocols, wherein selecting includes determining an authentication protocol appropriate for the message, wherein the authentication protocol selected is a function of whether the message was encrypted when received;

executing, at the application level proxy, the authentication protocol to authenticate the sender of the message; and

passing the decrypted message through the IP layer to its destination.

3. The method according to claim **2** wherein determining an authentication protocol appropriate for the message includes:

determining a source IP address associated with the message; and

determining the authentication protocol associated with the source IP address.

4. A method of authenticating the sender of a message within a computer system having a network protocol stack, wherein the network protocol stack includes an Internet Protocol (IP) layer, the method comprising:

determining, at the IP layer, if the message is encrypted;

if the message is encrypted, decrypting the message, wherein decrypting the message includes executing a process at the IP layer to decrypt the message;

passing the decrypted message up the network protocol stack to an application level proxy;

determining an authentication protocol appropriate for the message wherein the message includes a security parameters index and wherein determining an authentication protocol appropriate for the message includes:

determining the authentication protocol associated with a dynamic IP address, wherein determining the authentication protocol includes looking up a security association based on the security parameters index;

determining a current address associated with the dynamic source IP address; and

binding the current address to the security parameters index;

executing, at the application level proxy, the authentication protocol to authenticate the sender of the message; and

passing the decrypted message through the IP layer to its destination.

5. The method according to claim **4** wherein the authentication protocol selected is a function of whether the message was encrypted when received.

6. A firewall, comprising:

a first communications interface;

a second communications interface;

a network protocol stack connected to the first and the second communications interfaces, wherein the network protocol stack includes an Internet Protocol (IP) layer and a transport layer;

a decryption procedure, operating at the IP layer, wherein the decryption procedure decrypts encrypted messages received at one of said first and second communications interfaces and outputs decrypted messages; and

an application layer proxy, connected to the transport layer of said network protocol stack, wherein the application layer proxy includes a plurality of authentication protocols, wherein each authentication protocol provides a different level of security, wherein the application layer proxy receives decrypted messages from the decryption procedure, selects an authentication protocol from the plurality of authentication protocols as a function of content of the decrypted message and whether the message was encrypted when received, executes the selected authentication protocol, and returns the message to the IP layer.

7. A firewall, comprising:

a first communications interface;

a second communications interface;

a first network protocol stack connected to the first communications interface, wherein the first network protocol stack includes an Internet Protocol (IP) layer and a transport layer;

a second network protocol stack connected to the second communications interface, wherein the second network protocol stack includes an Internet Protocol (IP) layer and a transport layer;

a security policy;

a decryption procedure, operating at the IP layer of the first network protocol stack, the decryption procedure receiving encrypted messages received by said first communications interface and outputting decrypted messages; and

an application layer proxy, connected to the transport layers of said first and second network protocol stacks, wherein the application layer proxy includes a plurality of authentication protocols, wherein each authentication protocol provides a different level of security, wherein the application layer proxy receives decrypted messages from the decryption procedure, selects an authentication protocol from the plurality of authentication protocols based on content of the decrypted message and whether the message was encrypted when received, and executes the selected authentication protocol; and

wherein the application layer proxy determines based on the security policy whether the message is to be forwarded, and wherein the message is returned to the IP layer if the message is to be forwarded.

8. A firewall, comprising:

a first communications interface;

a second communications interface;

a first network protocol stack connected to the first communications interface, wherein the first network protocol stack includes an Internet Protocol (IP) layer and a transport layer;

a second network protocol stack connected to the second communications interface, wherein the second network protocol stack includes an Internet Protocol (IP) layer and a transport layer;

a security policy;

a decryption procedure, operating at the IP layer of the first network protocol stack, the decryption procedure receiving encrypted messages received by said first communications interface and outputting decrypted messages; and

an application layer proxy, connected to the transport layers of said first and second network protocol stacks, wherein the application layer proxy includes a plurality of authentication protocols, wherein each authentication protocol provides a different level of security, wherein the application layer proxy receives decrypted messages from the decryption procedure, selects an authentication protocol from the plurality of authentication protocols based on the content of the decrypted message, and executes the selected authentication protocol and wherein the application layer proxy determines based on the security policy whether the message is to be forwarded, and wherein the message is returned to the IP layer if the message is to be forwarded;

a third communications interface; and

a third network protocol stack connected to the third communications interface and to the application layer proxy, wherein the third network protocol stack includes an Internet Protocol (IP) layer and a transport layer and wherein the second and third network protocol stacks are restricted to first and second burbs, respectively.

9. The firewall according to claim **8** wherein the authentication protocol is also based on whether the message was encrypted when received.

10. A method of establishing a virtual private network between a first and a second network, wherein each network

includes an application level gateway firewall which uses a proxy operating at the application layer to process traffic through the firewall, wherein each firewall includes a network protocol stack and wherein each network protocol stack includes an Internet Protocol (IP) layer, the method comprising:

providing a plurality of authenication protocols, wherein each authentication prtocol provides a different level of security;

transferring a connection request from the first network to the second network;

determining, at the IP layer of the network protocol stack of the second network's firewall, if the connection request is encrypted;

if the connection request is encrypted, decrypting the request, wherein decrypting the request includes executing a procedure at the IP layer of the second network's firewall to decrypt the message;

passing the connection request up the network protocol stack to an application level proxy;

selecting an authentication protocol from the plurality of authentication protocols, wherein selecting includes determining an authentication protocol appropriate for the connection request, wherein the authentication protocol selected is a function of whether the message was encrypted when received;

executing the authentication protocol at the application level proxy to authenticate the connection request; and

if the connection request is authentic, establishing an active connection between the first and second networks and returning the connection request to the IP layer.

11. The method according to claim **10** wherein executing the authentication protocol includes executing program code within the firewall of the second network to mimic a challenge/response protocol executing on a server internal to the second network.

12. The method according to claim **10** wherein executing the authentication protocol includes executing program code to execute the authentication protocol in line to the session.

13. A computer-readable medium having computer-executable instructions for regulating the flow of messages between an external network and an internal network through a firewall having a network protocol stack, wherein the network protocol stack includes an Internet Protocol (IP), layer the instructions comprising:

instructions for establishing a security policy;

instructions for determining, at the IP layer, if a message to an IP address is encrypted;

instructions for passing the unencrypted message up the network protocol stack to an application level proxy if the message to the IP address is not encrypted;

instructions for decrypting the message and passing the decrypted message up the network protocol stack to the application level proxy if the message to the IP address is encrypted, wherein decrypting the message includes executing a procedure at the IP layer to decrypt the message;

instructions for determining at the application level proxy and based on the security policy if the message to that IP address is one that can be forwarded, wherein the decision whether to forward is a function of whether the message was encrypted when received; and

instructions for passing the message from the application level proxy to its destination through the IP layer.

14. A computer-readable medium having computer-executable instructions for authenticating the sender of a message within a computer system having a network protocol stack, wherein the network protocol stack includes an Internet Protocol (IP) layer the instructions comprising:

instructions for providing a plurality of authentication protocols, wherein each authentication protocol provides a different level of security;

instructions for determining, at the IP layer, if the message is encrypted;

instructions for decrypting the message if the message is encrypted, wherein decrypting the message includes executing a process at the IP layer to decrypt the message;

instructions for passing the decrypted message up the network protocol stack to an application level proxy;

instructions for selecting an authentication protocol from the plurality of authentication protocols, wherein selecting includes determining an authentication protocol appropriate for the message, wherein the authentication protocol selected is a function of whether the message was encrypted when received;

instructions for executing, at the application level proxy, the authentication protocol to authenticate the sender of the message; and

instructions for passing the decrypted message through the IP layer to its destination.

15. A computer-readable medium having computer-executable instructions for establishing a virtual private network between a first and a second network, wherein each network includes an application level gateway firewall which uses a proxy operating at the application layer to process traffic through the firewall, wherein each firewall includes a network protocol stack and wherein each network protocol stack includes an Internet Protocol (IP) layer, the instructions comprising:

instructions for providing a plurality of authentication protocols, wherein each authentication protocol provides a different level of security;

instructions for transferring a connection request from the first network to the second network;

instructions for determining, at the IP layer of the network protocol stack of the second network's firewall, if the connection request is encrypted;

instructions for decrypting the request if encrypted, wherein decrypting the request includes executing a procedure at the IP layer of the second network's firewall to decrypt the message;

instructions for passing the connection request up the network protocol stack to an application level proxy;

instructions for selecting an authentication protocol from the plurality of authentication protocols, wherein selecting includes determining an authentication protocol appropriate for the connection request, wherein the authentication protocol selected is a function of whether the message was encrypted when received;

instructions for executing the authentication protocol at the application level proxy to authenticate the connection request; and

instructions for establishing, if the connection request is authentic, an active connection between the first and second networks and for returning the connection request to the IP layer.

16. A computer-readable medium having computer-executable instructions for authenticating the sender of a message within a computer system having a network protocol stack, wherein the network protocol stack includes an Internet Protocol (IP) layer, the instructions comprising:

instructions for determining, at the IP layer, if the message is encrypted;

instructions for decrypting the message if encrypted, wherein decrypting the message includes executing a process at the IP layer to decrypt the message;

instructions for passing the decrypted message up the network protocol stack to an application level proxy;

instructions for determining an authentication protocol appropriate for the message wherein the message includes a security parameters index and wherein determining an authentication protocol appropriate for the message includes:

determining the authentication protocol associated with a dynamic IP address, wherein determining the authentication protocol includes looking up a security association based on the security parameters index;

determining a current address associated with the dynamic source IP address; and

binding the current address to the security parameters index;

instructions for executing, at the application level proxy, the authentication protocol to authenticate the sender of the message; and

instructions for passing the decrypted message through the IP layer to its destination.

* * * * *

US006061797A

United States Patent [19]

Jade et al.

[11] **Patent Number: 6,061,797**

[45] **Date of Patent: May 9, 2000**

[54] **OUTSIDE ACCESS TO COMPUTER RESOURCES THROUGH A FIREWALL**

[75] Inventors: **Prashanth Jade**, Delray Beach; **Victor Stuart Moore**, Boynton Beach, both of Fla.; **Arun Mohan Rao**, Dallas, Tex.; **Glen Robert Walters**, Sebring, Fla.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **09/132,915**

[22] Filed: **Aug. 12, 1998**

Related U.S. Application Data

[63] Continuation of application No. 08/731,800, Oct. 21, 1996, Pat. No. 5,944,823.

[51] **Int. Cl.**^{7} **G06F 12/14**; G06F 13/16; H04L 29/06; H04L 9/32

[52] **U.S. Cl.** **713/201**; 709/229; 380/25

[58] **Field of Search** 709/218, 230, 709/217, 229, 749, 227, 207, 104; 345/331; 707/10, 9, 6, 8; 382/115; 235/380; 713/201; 380/23; 710/200; 350/201, 25; 711/203, 163

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,283,828	2/1994	Sauders et al.	380/4
5,455,953	10/1995	Russell	395/739
5,481,715	1/1996	Hamilton et al.	395/700
5,548,646	8/1996	Aziz et al.	380/23
5,826,014	10/1998	Colly et al.	713/201

FOREIGN PATENT DOCUMENTS

WO 97 16911	5/1997	WIPO .

OTHER PUBLICATIONS

Cheswick and Bellovin: "Firewalls and Internet Security, repelling the Willy Hacker"; Apr. 1994, Addison–Wesley Publishing Company; pp. 86 to 106.

Bryan J: "Firewalls for Sale"; BYTE, vol. 20, No. 4, Apr. 1, 1995; pp. 99/100, 102, 104.

Ted Doty: "A firewall Overview"; CONNEXIONS, vol. 9, No. 7, Jul. 1, 1995; pp. 20–23.

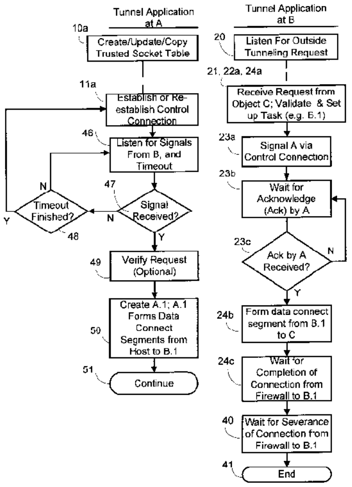
(List continued on next page.)

Primary Examiner—Daniel H. Pan
Attorney, Agent, or Firm—Richard A. Tomlin; Robert Lieber

[57] **ABSTRACT**

A firewall isolates computer and network resources inside the firewall from networks, computers and computer applications outside the firewall. Typically, the inside resources could be privately owned databases and local area networks (LAN's), and outside objects could include individuals and computer applications operating through public communication networks such as the Internet. Usually, a firewall allows for an inside user or object to originate connection to an outside object or network, but does not allow for connections to be generated in the reverse direction; i.e. from outside in. The disclosed invention provides a special "tunneling" mechanism, operating on both sides of a firewall, for establishing such "outside in" connections when they are requested by certain "trusted" individuals or objects or applications outside the firewall. The intent here is to minimize the resources required for establishing "tunneled" connections (connections through the firewall that are effectively requested from outside), while also minimizing the security risk involved in permitting such connections to be made at all. The mechanism includes special tunneling applications, running on interface servers inside and outside the firewall, and a special table of "trusted sockets" created and maintained by the inside tunneling application. Entries in the trusted sockets table define objects inside the firewall consisting of special inside ports, a telecommunication protocol to be used at each port, and a host object associated with each port. Each entry is "trusted" in the sense that it is supposedly known only by individuals authorized to have "tunneling" access through the firewall from outside.

2 Claims, 2 Drawing Sheets


Tunnel Application at A
10a
Create/Update/Copy Trusted Socket Table
11a
Establish or Re-establish Control Connection
46
Listen for Signals From B, and Timeout
47
Signal Received?
N
Timeout Finished?
N
Y
48
Y
49
Verify Request (Optional)
50
Create A.1; A.1 Forms Data Connect Segments from Host to B.1
51
Continue
Tunnel Application at B
20
Listen For Outside Tunneling Request
21, 22a, 24a
Receive Request from Object C; Validate & Set up Task (e.g. B.1)
23a
Signal A via Control Connection
23b
Wait for Acknowledge (Ack) by A
23c
Ack by A Received?
N
Y
24b
Form data connect segment from B.1 to C
24c
Wait for Completion of Connection from Firewall to B.1
40
Wait for Severance of Connection from Firewall to B.1
41
End

OTHER PUBLICATIONS

Bellovin S M et al: "Network Firewalls" IEEE Communications Magazine, vol. 32, No. 9, Sep. 1, 1994, pp. 50–57B.

Newman D et al: Can Firewalls Take the Heat?; Data Communications, vol. 24, No. 16, Nov. 21, 1995; pp. 71–78, 80.

Noritoshi Demizu et al: "DDT—A Versatile Tunneling Technology"; Computer Networks and ISDN Systems, vol. 27, No. 3, Dec. 1, 1994, pp. 493–502.

PCT International Preliminary Examination Report, Oct. 2, 1997, International Application No. PCT/GB97/02712.

Fig. 1

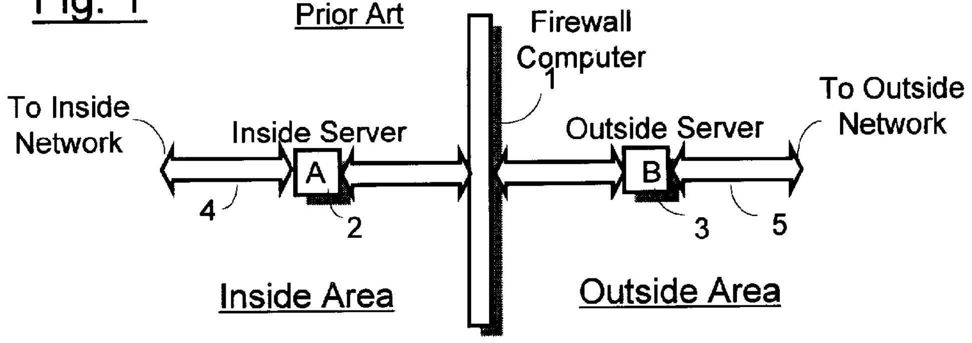


Fig. 2

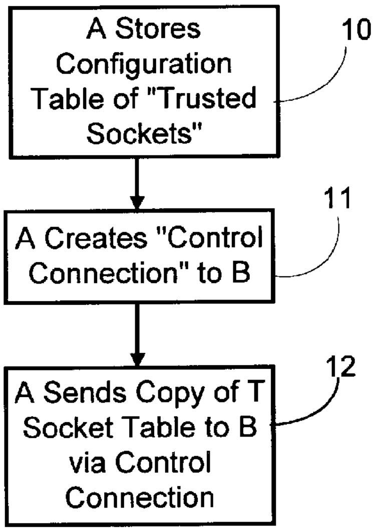
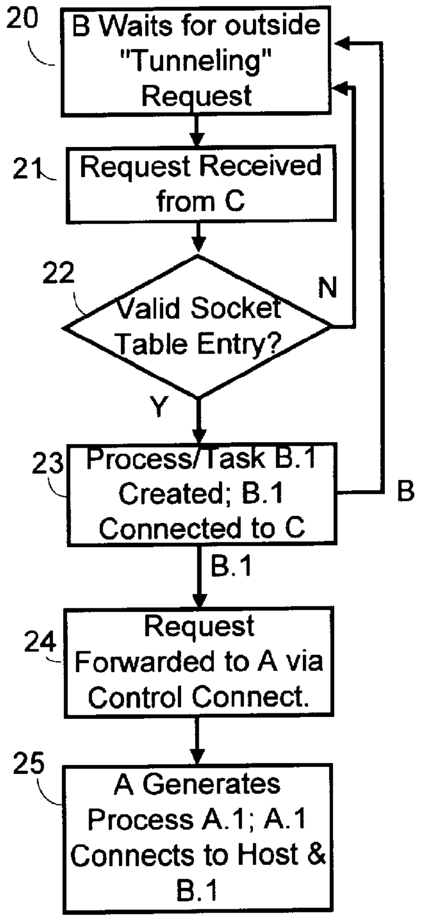

A Stores Configuration Table of "Trusted Sockets"
10
A Creates "Control Connection" to B
11
A Sends Copy of T Socket Table to B via Control Connection
12

Fig. 3

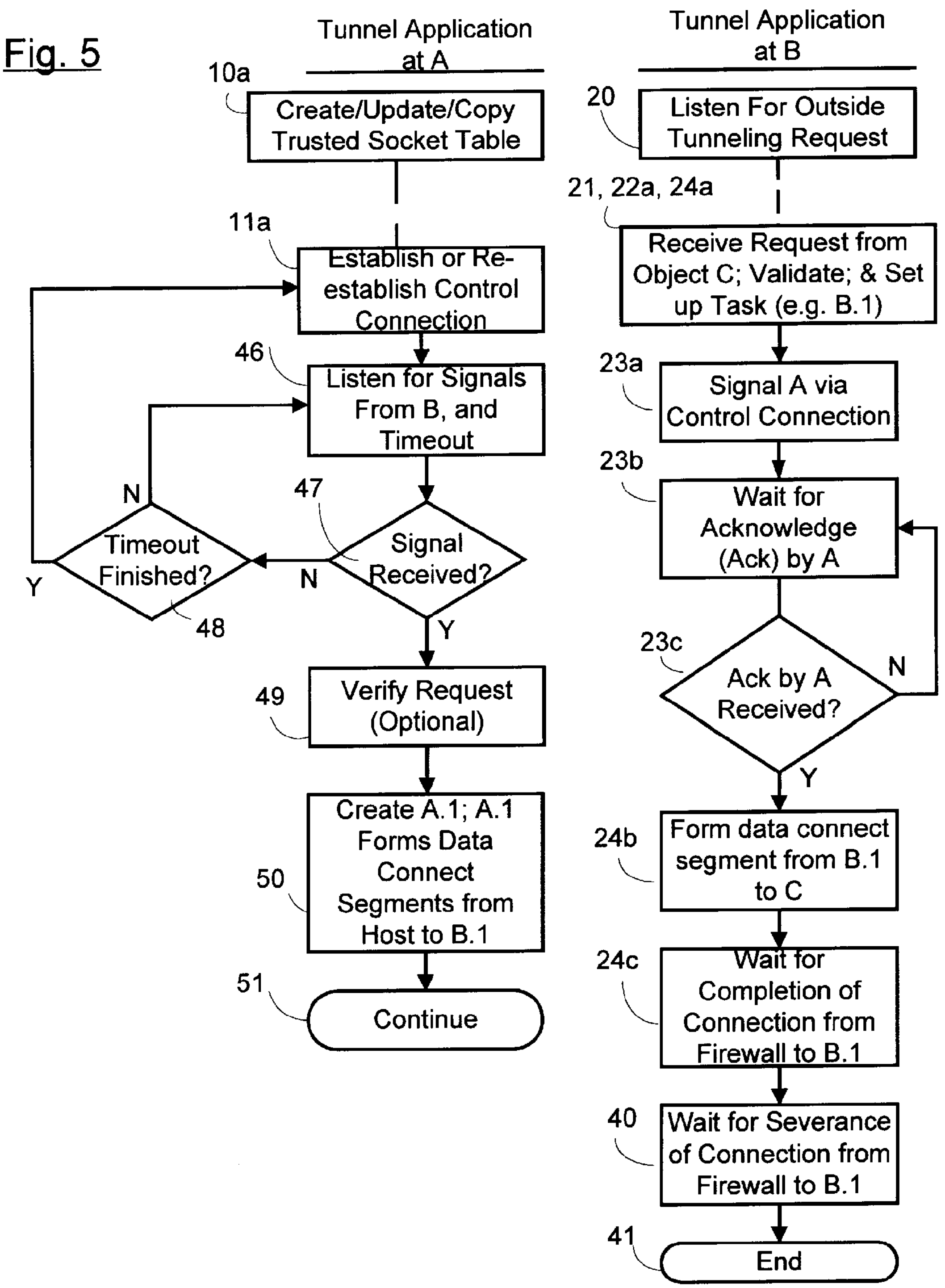

20
B Waits for outside "Tunneling" Request
21
Request Received from C
22
Valid Socket Table Entry?
N
Y
23
Process/Task B.1 Created; B.1 Connected to C
B
B.1
24
Request Forwarded to A via Control Connect.
25
A Generates Process A.1; A.1 Connects to Host & B.1

[Result: Host and C are connected (bidirectionally)]

Fig. 4

"Trusted Socket Port"	Protocol	Host ID	
118	NNTP	VVU	30
119	HTTP	XYZ	
			31

Fig. 5


Tunnel Application at A
10a
Create/Update/Copy Trusted Socket Table
11a
Establish or Re-establish Control Connection
46
Listen for Signals From B, and Timeout
47
Signal Received?
N
Y
Timeout Finished?
N
Y
48
49
Verify Request (Optional)
50
Create A.1; A.1 Forms Data Connect Segments from Host to B.1
51
Continue
Tunnel Application at B
20
Listen For Outside Tunneling Request
21, 22a, 24a
Receive Request from Object C; Validate; & Set up Task (e.g. B.1)
23a
Signal A via Control Connection
23b
Wait for Acknowledge (Ack) by A
23c
Ack by A Received?
N
Y
24b
Form data connect segment from B.1 to C
24c
Wait for Completion of Connection from Firewall to B.1
40
Wait for Severance of Connection from Firewall to B.1
41
End

OUTSIDE ACCESS TO COMPUTER RESOURCES THROUGH A FIREWALL

This is a continuation under 37 CFR 1.53(b) of parent patent application Ser. No. 08/731,800 filed Oct. 21, 1996, now U.S. Pat. No. 5,944,823. The disclosure of that parent application is hereby incorporated into this application by reference.

FIELD OF THE INVENTION

This invention concerns provision of access to resources of a computer system or network, to objects outside a security firewall, in response to requests from respective objects.

BACKGROUND OF THE INVENTION

A firewall is a security system (hardware and/or software) that isolates resources of a computer system or network from objects outside of the system or network. Isolated resources are characterized as inside the firewall, and external equipment is considered outside the firewall. Typically, a firewall serves as a security enclosure around a private local area network (LAN) of computers and associated peripherals.

Generally, a firewall allows for inside objects to request and receive connections to outside objects (e.g. for inside applications to access outside internet nodes, etc.), but prevents outside objects from originating similar connections.

There are instances where it is desired to allow for objects outside a firewall to have access to inside resources, subject to restrictions that would not fully defeat the security purpose of the firewall. For example, it might be desirable to allow employees of a company owning resources inside the firewall to "telecommute" over public networks (such as the telephone network or that network and Internet points of access, etc.), from homes remote from their employer's place(s) of business (or from remote locations while on business trips or vacations). For that purpose then it would be desirable to permit such "trusted" individuals to be able to initiate access outside a firewall to resources inside the firewall (e.g. the employer's private databases, etc.).

To our knowledge, such access, in response to outside initiation or request, has been provided in the past by providing duplicate servers and database stores, both inside and outside the firewall, or by means of other arrangements that add considerable expense to maintenance of the firewall per se. Consider, for example, the costs of such outside duplication, or other process, in relation to massive and frequently updated databases stored inside the firewall. The present invention seeks to provide the desired outside access without unnecessary outside duplication of objects or resources inside the firewall.

SUMMARY OF THE INVENTION

In accordance with the invention, means are provided inside and outside a firewall for cooperatively producing tunneling effects, in response to certain types of requests initiated by objects outside the firewall, which effects result in creation of connections between such outside objects and resources inside the firewall. Connections so created have the unique property that they are effectively created from "inside out" as if they were requests originating from objects inside the firewall to destinations outside the firewall.

The "types of requests" accommodated by such "tunneling" means are requests addressed to what are presently termed "trusted sockets". Trusted sockets are entries in a table of trusted sockets that is created and maintained exclusively inside the firewall. Each entry in that table includes the address of a "trusted" port, a protocol (e.g. a telecommunication protocol such as TCP/IP, NNTP, etc.) pre-associated with that address, and the identity of a host object inside the firewall (e.g. a host computer or a host application). Thus, it is understood that in order for an individual and/or object outside the firewall to initiate such a request that individual must be entrusted with the information that represents a trusted socket entry that is currently valid.

The table of trusted sockets is created and maintained by a "tunneling application" running on an inside interface server (under control of appropriately authorized individuals having direct access to that server) that interfaces between this tunneling application and all other "accessible" objects/resources inside the firewall (including other applications running on the inside interface server). The inside interface server also establishes a "control connection" to an outside interface server which interfaces between the firewall and all objects outside the firewall. The control connection is accessible only to the tunneling application running on the inside interface server and a corresponding tunneling application running on the outside interface server; i.e. it is not directly accessible to any other applications running on these interfaces servers, and is totally inaccessible to both inside and outside objects not residing on these servers.

A copy of the trusted sockets table is transferred from the inside interface server to the outside interface server; e.g. when the table is created and/or altered, or at special times of day, etc.

When an outside object, that is currently not connected through the firewall, originates a request reaching the outside interface server, the tunneling application on that server determines if the request is directed to a trusted socket entry that is currently valid. If it is not so directed, the request is ignored. If the request is to a trusted socket, the request is passed over the control connection to the tunneling application on the inside interface server. Concurrently, a process (or task) associated with the request is generated in the outside interface server, and an outside connection is established between that process/task and the requesting object.

Upon receiving the request, the inside tunneling application also may be required to verify that the request is to a currently valid trusted socket and disallow the request if it is not. If the request is to a currently valid trusted socket, the inside tunneling application generates (or "spawns") an inside process associated with the request. Then the inside tunneling application: (a) generates connections between the inside resource associated with the port and host identity of the "requested" trusted socket entry and the inside interface server; and (b) communicating over the control connection with the outside tunneling application and a computer controlling the firewall itself, generates a connection through the firewall between the tasks generated/spawned on both the inside and outside interface servers. The connections generated/spawned by the inside and outside tunneling applications are separate from the control connection, and useful to carry data (usually in packet format defined by the trusted socket protocol) bidirectionally between the outside object that originated the request and the inside object targeted by the request.

These and other features, advantages, objectives and benefits of the present invention will be more fully understood by considering the following detailed description and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a schematic of a typical firewall environment in which the present invention can be applied.

FIG. **2** is a flow diagram illustrating the creation and handling of the trusted socket table mentioned above.

FIG. **3** is a flow diagram illustrating the firewall tunneling process of the present invention.

FIG. **4** illustrates a preferred form of the trusted sockets table mentioned above.

FIG. **5** is a flow diagram for explaining details of tunneling application operations inside and outside of a firewall, in accordance with the present invention.

DETAILED DESCRIPTION

FIG. **1** illustrates a typical firewall environment for application of the present invention. Firewall computer **1** maintains the firewall security function in accordance with presently commonplace procedures. The functions of this computer, other than those involving extending connections from objects inside the firewall to objects outside the firewall, are transparent to (and in essence not relevant to) the present invention. Interface servers **2** and **3** (labelled servers A and B respectively) operate respectively inside and outside the firewall created by **1**. Server A interfaces between the firewall and objects (software applications, hardware entities, etc.) inside the firewall, including objects in Server A itself. Server B interfaces between the firewall and objects outside the firewall, including objects in server B itself.

In a typical firewall usage environment, server A connects to a network inside the firewall (e.g. a private local area network) via a connection shown at **4**, and server B connects to a network outside the firewall (e.g. the Internet) via a connection shown at **5**.

In applying the present invention to this environmental configuration, servers A and B are provided with "tunneling" software applications and store copies of a "trusted socket" table. These entities—the tunneling applications and the trusted socket table—are considered unique to the present invention and described herein.

FIGS. **2** and **3** describe (tunneling) processes performed at servers A and B in furtherance of the present invention.

As shown at **10**, in FIG. **2**, a trusted socket table (which is described below in reference to FIG. **4**) is created in and stored at server A (or a store readily accessible to that server). As shown at **11**, server A creates a special "control connection" to server B through the firewall (computer), and passes a copy of the trusted sockets table to server B over the control connection. This control connection, also considered part of the present invention, is used by the above-mentioned tunneling applications to effectively inter-communicate, and thereby form other connections (hereinafter termed "data connections") between objects inside and outside the firewall, in response to requests received from outside objects.

Segments of these data connections extending through the firewall are entirely separate from the control connection used in their formation, and are always formed under control of processes running inside the firewall. For an outside request to give rise to formation of a data connection to an inside object, the request must be directed to an entry in the trusted sockets table, and validated as such. Outside requests found to be invalid are ignored, so that the firewall and its inside resources are effectively invisible to and inaccessible to outside requestors having invalid request information. Conversely, it should be understood that valid requests are issuable only at the direction of individuals having privileged knowledge of currently valid entries in the trusted sockets table (e.g. telecommuting employees of the owner of the inside resources, etc.).

FIG. **3** describes tunneling functions performed at servers A and B, after B has received and stored its copy of the trusted sockets table sent by A.

As shown at **20**, (the tunneling application in) server B waits to receive an outside request that effectively calls for a tunneling operation; i.e. creation of a data connection between an inside "host" object designated in the request and the outside object from which the request was sent. Upon receiving a request (**21**, FIG. **3**), (the tunneling application at) B checks to verify that the request is a valid one (decision **22**, FIG. **3**). In respect to the last-mentioned function, it should be understood that server B only receives requests directed to that server, and that the tunneling application on server B only receives requests that appear to be directed to a port inside the firewall, and distinguishes those requests as valid only if they are directed to a currently valid entry in the trusted sockets table mentioned earlier.

If the request is invalid it is ignored, and (the application at) server B resumes waiting for a request. However, if the request is valid, (the tunneling application at) server B creates a process or task "B.1" for handling outside elements of data transfer relative to the requesting object (**23**, FIG. **3**). Task B.1 establishes a data connection between itself and the requesting object (also **23**, FIG. **3**), and forwards the request to (the tunneling application at) server A_1 via the control connection, along with the identity of task B.1 (**24**, FIG. **3**).

Upon receiving a validated request, (the tunneling application at) server A generates a process or task A.1, for handling inside aspects of the transmission of data between the outside requesting object and a host object identified in the request (**25**, FIG. **3**; the latter object being a component of a trusted socket designation as explained below). Task A.1 creates data connection segments from the host object to the firewall computer (also **25**, FIG. **3**), and instructs the firewall computer to form a connection to B.1 (also **25**, FIG. **1**); thus completing a data connection between the inside host object and the outside requesting object. It should be appreciated that this data connection may require buffers, in servers A and B and the firewall computer, of a size determined by the protocol of data transmission (discussed further below), and the required speed of (packet) transfer for that protocol.

The form of the trusted sockets table is illustrated in FIG. **4**. Examples of 2 specific entries are shown at **30**, and additional entries are implied at **31** by dotted lines extending downward from the second entry. Each entry consists of a port number, information defining a transmission protocol (usually, a burst packet transfer protocol), and information identifying a host object. The port number is an address inside the firewall assigned to the host object. As examples of protocols, the first two entries in the table list NNTP (Network News Transport Protocol) and HTTP (HyperText Transport Protocol).

FIG. **5** shows in finer detail operations performed by the tunneling applications at interface servers A and B. Operations that are the same as operations shown in FIGS. **2** and **3** are identified by identical numerals. Operations that are parts of, or differ in some respect from, operations shown in FIGS. **2** and **3** are identified by the same numbers followed by letters (a, b, etc.). Other operations are identified by numbers different from those previously used.

Operation **10***a* at server A, a composite of operations **10** and **12** of FIG. **2**, is the creation and updating (expansion, modification. etc.) of the trusted sockets table and the copying of the latter to server B. Operation **11***a* at server A is the establishment or (as explained below) re-establishment of the control connection between (the tunneling applications at) servers A and B. A need to reestablish the control connection arises when the connection is unintentionally broken, and the operations required to detect and respond to such occurrences are shown at **46–48** in FIG. **5** (which are discussed further below).

After receiving its copy of the trusted sockets table, (the tunneling application at) server B listens for outside requests (**20**, FIG. **5**). When a valid outside tunneling request is received, and an associated data handling task (e.g. B.1, FIG. **3**) has been created therefor (**21–22***a*, **24***a*, FIG. **5**), server B presents the request to server A (**23***a*, FIG. **5**), along with control signals indicating the action occurring and information identifying the task (e.g. B.1) created at B to attend to the request. Server B then waits for acknowledgement of receipt of the request from server A (**23***c*, FIG. **5**), and upon receiving such server B establishes a data connection segment from the newly created task to the requesting object (**24***b*, FIG. **5**; e.g. from B.1 to C as in FIG. **3**). Server B then waits for establishment of a data connection segment from the firewall to the task just created at B (**24***c*, FIG. **5**), that occurrence implying establishment of an associated data connection segment between the host object (the one identified in the request) and server B. The tunneling process at server B is then complete until the data connection segment between the firewall and the task at B is terminated (**40**, FIG. **5**), ending the involvement of server B in that connection and the associated request (**41**, FIG. **5**).

Returning to consideration of tunneling actions at server A, after establishing or re-establishing the control connection, server A listens for (request forwarding) signals from B (**46**, FIG. **5**). If a signal hasn't been received (**47**, FIG. **5**), but a predetermined timeout interval has not elapsed since the waiting started (**48**, FIG. **5**), server A merely continues to wait for such signal. However, if the timeout has lapsed (Yes decision at **48**, FIG. **5**) it is assumed that the control connection has been (unintentionally) broken, and the connection is re-established (**11***a* repeated).

If a request is received from server B, server A may optionally perform its own validation operation (**49**, FIG. **5**) to verify that the request is to a currently valid trusted socket. If that option is used and the request is found to be invalid, an error signal would be returned to server B instead of the acknowledgement awaited at **23***b*. If the option is not used, or if it is used and the request is found to be valid, server A proceeds to establish its internal task such as A.1, and the latter, as described previously, forms data connection segments from the host object to the firewall, and directs the firewall computer to extend the data connection to B.1 (**50**, FIG. **5**). This concludes server A's involvement in the current request, freeing it to continue with other requests (**51**, FIG. **5**).

Program Products

The foregoing tunneling applications can be delivered as a "computer readable" program product; e.g. on storage media or through communication networks. It should be understood that such product can be provided as either a single integral entity (e.g. one installed on inside server A and transferred in whole or part to outside server B), or two entities (or parts) separately installable on inside and outside servers. It also should be understood that the firewall computer is a necessary participant in the creation of data connections through the firewall.

Accordingly, we claim:

1. A tunneling software application contained on computer-readable media, said tunneling application being useful, in first and second computers connecting directly to opposite sides of a network security firewall and respectively connecting said opposite sides to first and second data communication networks, for isolating secure objects in said first network from said second network, while allowing for transfer of data through said firewall and between said first and second networks without compromising security of said secure objects in said first network; said tunneling application comprising:

first and second program segments respectively intended to run on said first and second computers; said first program segment operating to provide a data communication interface between said firewall and said first network, and said second program segment operating to provide a data communication interface between said firewall and said second network;

said first program segment comprising means for operating said first computer to create and maintain a table of trusted objects including objects associated with said secure objects in said first network; and means for operating said first computer, relative to said firewall and said second computer, to provide a copy of said table to said second computer for use by said second program segment.

2. A tunneling application in accordance with claim **1** wherein:

said first program segment comprises means for operating said first computer relative to said firewall said second computer to establish a private control connection through said firewall between said first and second computers; said private control connection being normally inaccessible to communications directed from said second network to said first network; and

said second program segment comprises means for operating said second computer to receive a communication directed from said second network to said first network, verify that said received communication is addressed to a said trusted object designated in said copy of said table of trusted objects, and forward data contained in said received communication to said first program segment in said first computer through said private control connection; in a form enabling said first program segment to have said contained data delivered to a said secure object without exposing the respective secure object to access from said second network.

* * * * *

US006651174B1

(12) **United States Patent**
Nagaoka et al.

(10) **Patent No.: US 6,651,174 B1**
(45) **Date of Patent: Nov. 18, 2003**

(54) **FIREWALL PORT SWITCHING**

(75) Inventors: **Toru Nagaoka**, Tokyo (JP); **Masashi Sakata**, Tokyo (JP); **Kazue Kobayashi**, Tokyo (JP)

(73) Assignee: **NTT Comware Corporation** (JP)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/274,384**

(22) Filed: **Mar. 23, 1999**

(30) **Foreign Application Priority Data**

May 27, 1998 (JP) 10-146372

(51) **Int. Cl.**^{7} **H04L 9/00**

(52) **U.S. Cl.** **713/201**; 709/239

(58) **Field of Search** 713/200, 201, 713/202, 150, 168, 169; 710/12; 709/223, 224, 225, 227, 228, 230, 238, 239, 250; 703/25; 370/351, 357, 359

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,734,865	A	*	3/1998	Yu	709/250
5,852,721	A	*	12/1998	Dillon et al.	709/217
5,968,129	A	*	10/1999	Dillon et al.	709/233
5,995,725	A	*	11/1999	Dillon	709/203
6,088,728	A	*	7/2000	Bellemore et al.	709/227
6,098,172	A	*	8/2000	Coss et al.	713/201
6,104,716	A	*	8/2000	Crichton et al.	370/401
6,141,749	A	*	10/2000	Coss et al.	713/162
6,154,775	A	*	11/2000	Coss et al.	709/225
6,170,012	B1	*	1/2001	Coss et al.	709/229
6,185,598	B1	*	2/2001	Farber et al.	709/200
6,195,366	B1	*	2/2001	Kayashima et al.	370/475
6,202,157	B1	*	3/2001	Brownlie et al.	713/201
6,253,751	B1	*	7/2001	Carlsson	123/683
6,321,259	B1	*	11/2001	Ouellette et al.	709/220
2002/0073338	A1	*	6/2002	Burrows et al.	713/201

OTHER PUBLICATIONS

Wong, "Serving up digital certificates" May 1, 1998, Network, Dialog text search, p. 1–6.*

Rutrell, "VPN authentication moves to LANs– Alcatel adds Radius technology, typically used for remote access, to its switch" 2000, Internetweek, #810, Dialog text search, p. 1.*

Freier et al, "The SSL Protocol Version 3.0" Nov. 18, 1996, Transport Security Working Group, p. 1–67.*

* cited by examiner

Primary Examiner—Gail Hayes
Assistant Examiner—Christopher Revak
(74) *Attorney, Agent, or Firm*—Burns, Doane, Swecker & Mathis, LLP

(57) **ABSTRACT**

The present invention relates to a network system, the network system of the present invention comprises an authorized client terminal which is connected to a network, a server which is connected to the network, and a firewall which is interposed between the server and the network. The client terminal accesses the server by means of a publicly known protocol via a port having a publicly known port number in the firewall. In the case in which the accessing client terminal is authorized, the server downloads program for realizing effective dedicated protocols solely between the client terminal and itself to the client terminal via the port having the publicly known port number. Furthermore, the server access with the client terminal conducts data communication by executing the program and by means of the dedicated protocols, via the network and the port having the publicly known port number.

6 Claims, 8 Drawing Sheets

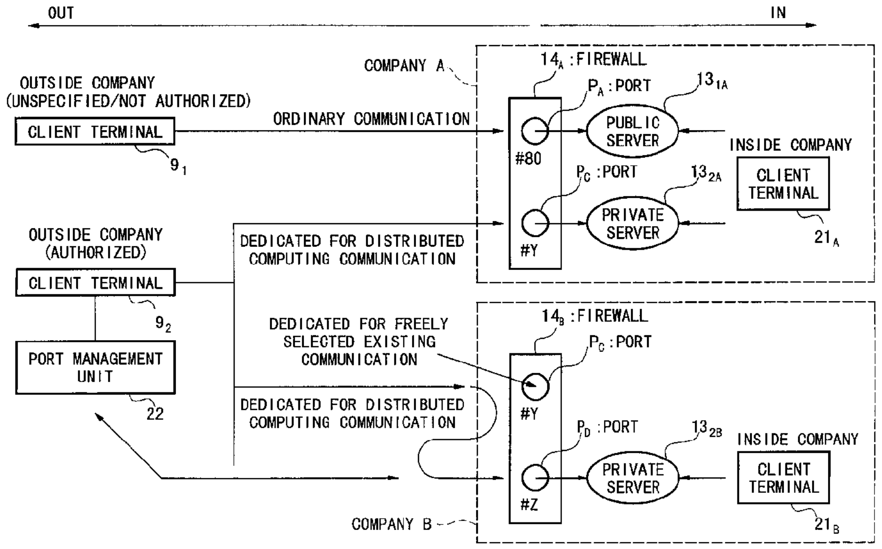


FIG.1

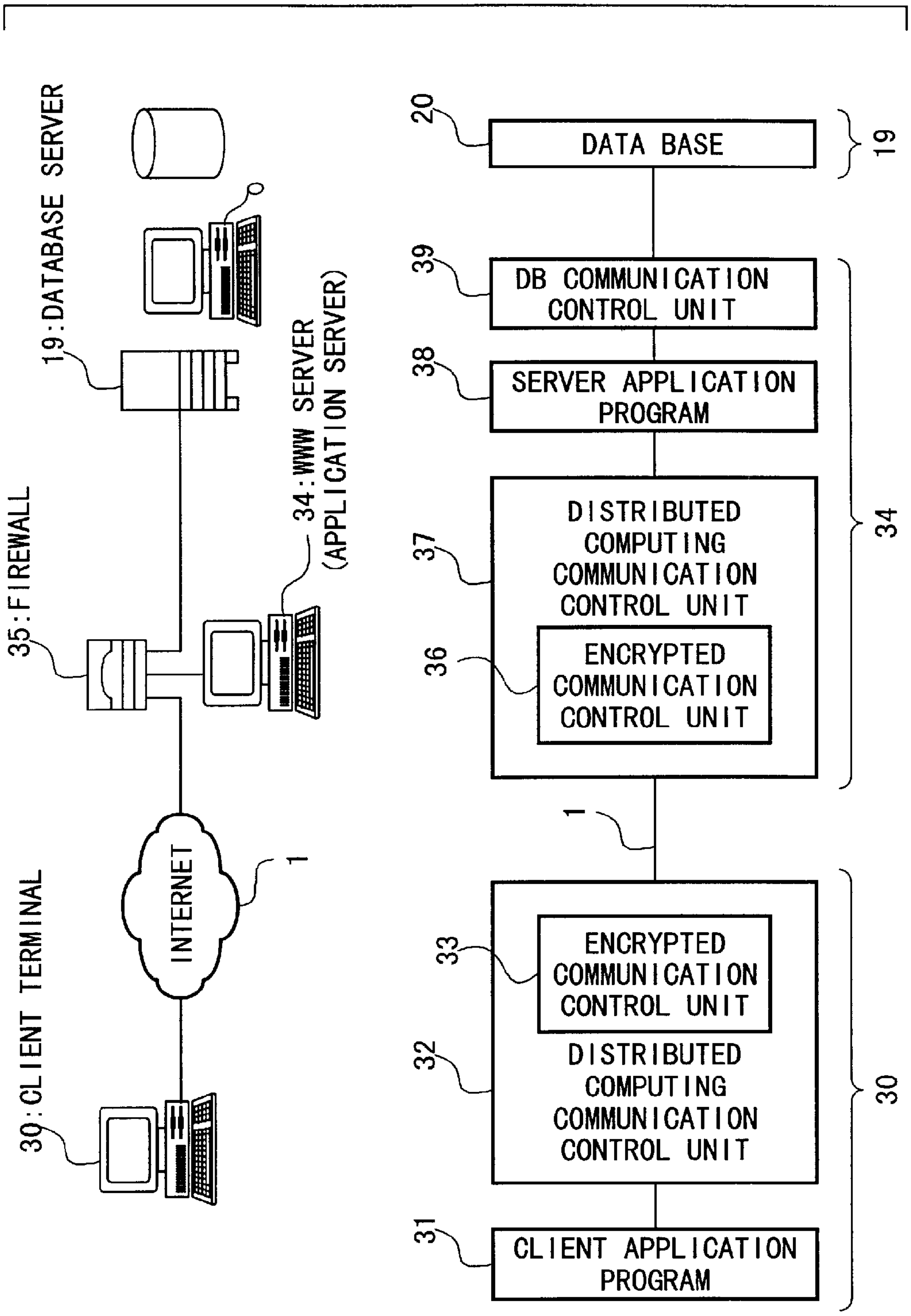

19:DATABASE SERVER
34:WWW SERVER (APPLICATION SERVER)
35:FIREWALL
INTERNET
1
30:CLIENT TERMINAL
20
DATA BASE
19
39
DB COMMUNICATION CONTROL UNIT
38
SERVER APPLICATION PROGRAM
37
DISTRIBUTED COMPUTING COMMUNICATION CONTROL UNIT
36
ENCRYPTED COMMUNICATION CONTROL UNIT
34
1
33
ENCRYPTED COMMUNICATION CONTROL UNIT
32
DISTRIBUTED COMPUTING COMMUNICATION CONTROL UNIT
30
31
CLIENT APPLICATION PROGRAM

FIG.2

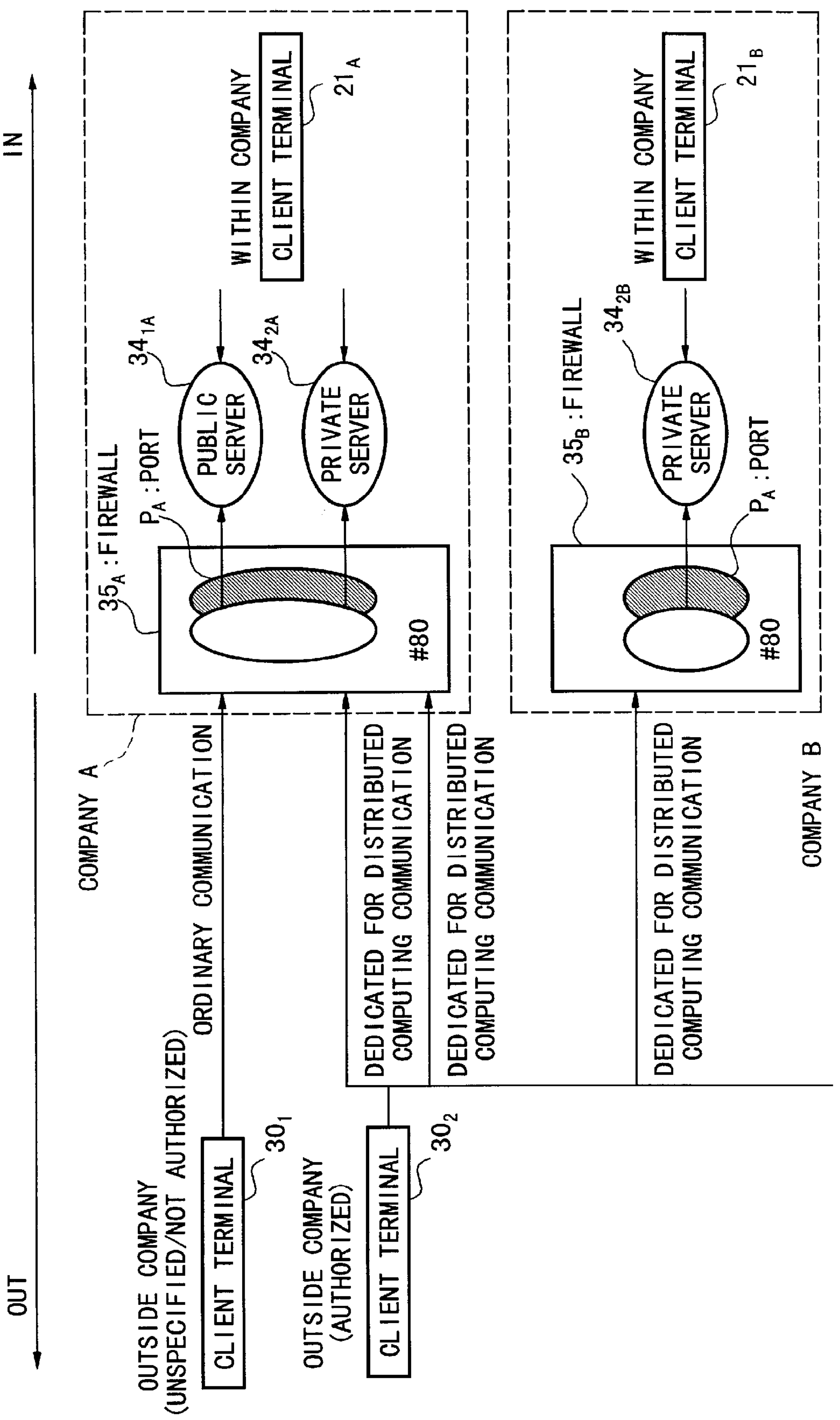

IN
OUT
COMPANY A
35A : FIREWALL
PA : PORT
#80
PUBLIC SERVER
34 1A
PRIVATE SERVER
34 2A
WITHIN COMPANY
CLIENT TERMINAL
21A
OUTSIDE COMPANY (UNSPECIFIED/NOT AUTHORIZED)
CLIENT TERMINAL
30 1
ORDINARY COMMUNICATION
OUTSIDE COMPANY (AUTHORIZED)
CLIENT TERMINAL
30 2
DEDICATED FOR DISTRIBUTED COMPUTING COMMUNICATION
DEDICATED FOR DISTRIBUTED COMPUTING COMMUNICATION
DEDICATED FOR DISTRIBUTED COMPUTING COMMUNICATION
COMPANY B
35B : FIREWALL
PA : PORT
#80
PRIVATE SERVER
34 2B
WITHIN COMPANY
CLIENT TERMINAL
21B

FIG.3

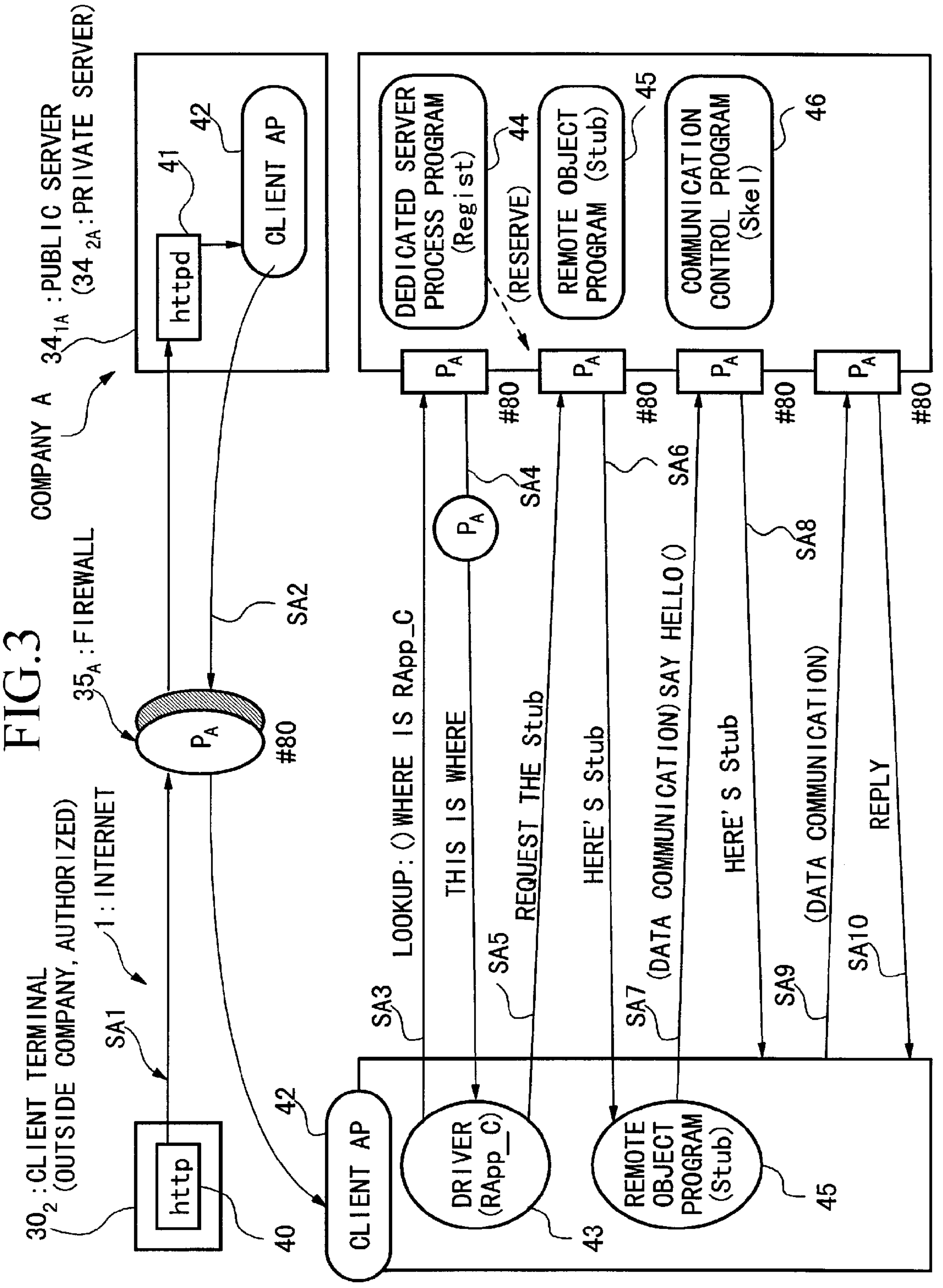

30_2 : CLIENT TERMINAL (OUTSIDE COMPANY, AUTHORIZED)
http
40
SA1
1 : INTERNET
35_A : FIREWALL
P_A
#80
SA2
COMPANY A
34_{1A} : PUBLIC SERVER (34_{2A} : PRIVATE SERVER)
httpd
41
CLIENT AP
42
DEDICATED SERVER PROCESS PROGRAM (Regist)
44
(RESERVE)
REMOTE OBJECT PROGRAM (Stub)
45
COMMUNICATION CONTROL PROGRAM (Skel)
46
P_A
#80
P_A
#80
P_A
#80
P_A
#80
SA4
P_A
SA6
SA8
LOOKUP: ( ) WHERE IS RApp_C
THIS IS WHERE
REQUEST THE Stub
HERE'S Stub
(DATA COMMUNICATION) SAY HELLO ( )
HERE'S Stub
(DATA COMMUNICATION)
REPLY
SA3
SA5
SA7
SA9
SA10
CLIENT AP
42
DRIVER (RApp_C)
43
REMOTE OBJECT PROGRAM (Stub)
45

FIG.4

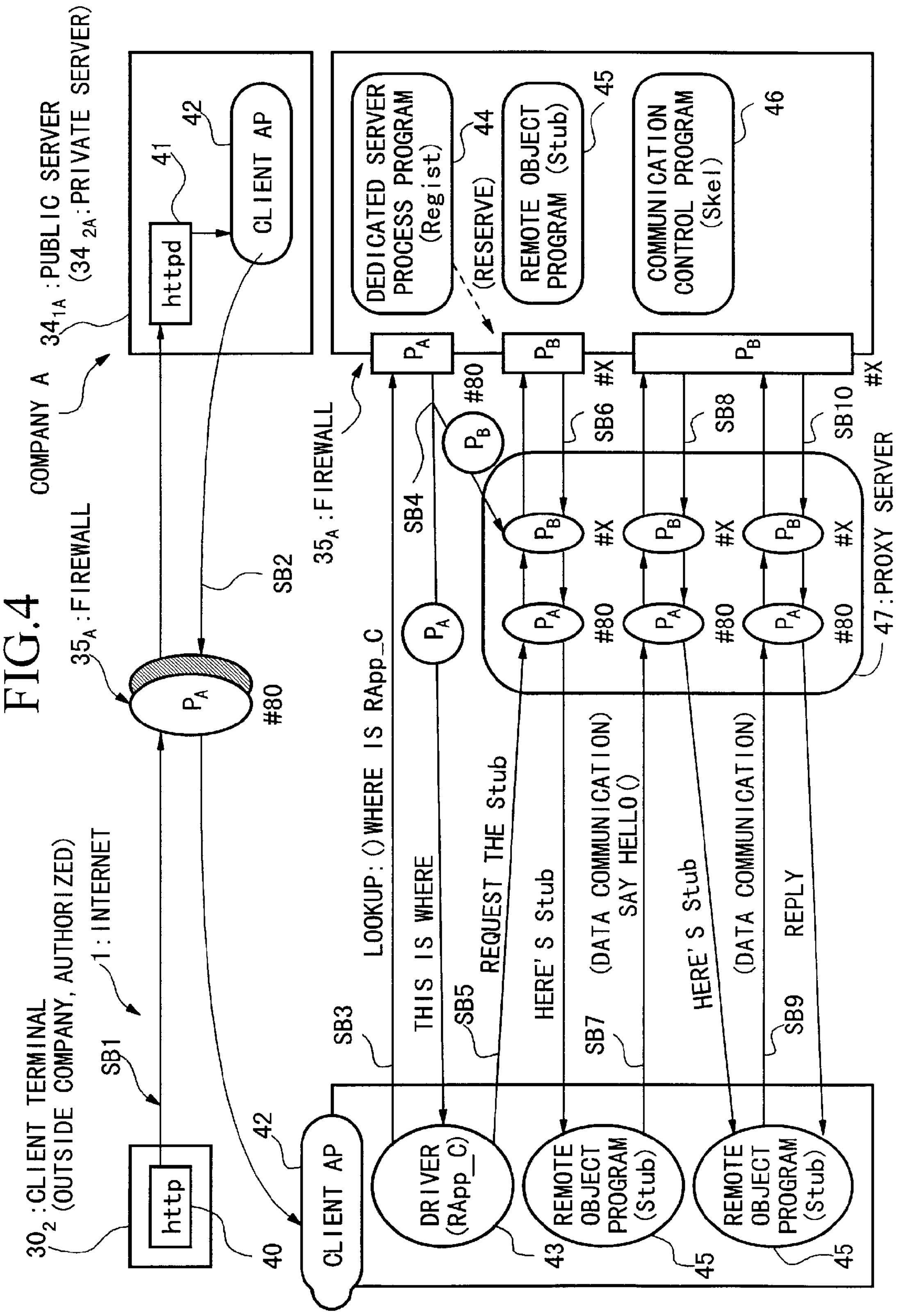

30₂ :CLIENT TERMINAL (OUTSIDE COMPANY, AUTHORIZED)
SB1
1:INTERNET
35A :FIREWALL
COMPANY A
34 1A :PUBLIC SERVER (34 2A:PRIVATE SERVER)
http
40
PA
#80
SB2
httpd
41
CLIENT AP
42
35A :FIREWALL
CLIENT AP
42
DRIVER (RApp_C)
43
REMOTE OBJECT PROGRAM (Stub)
45
REMOTE OBJECT PROGRAM (Stub)
45
SB3
LOOKUP:()WHERE IS RApp_C
THIS IS WHERE
SB4
SB5
REQUEST THE Stub
HERE'S Stub
SB6
SB7
(DATA COMMUNICATION) SAY HELLO()
HERE'S Stub
SB8
SB9
(DATA COMMUNICATION)
REPLY
SB10
PA
PB
#80
#X
47:PROXY SERVER
DEDICATED SERVER PROCESS PROGRAM (Regist)
44
(RESERVE)
REMOTE OBJECT PROGRAM (Stub)
45
COMMUNICATION CONTROL PROGRAM (Skel)
46

FIG.5

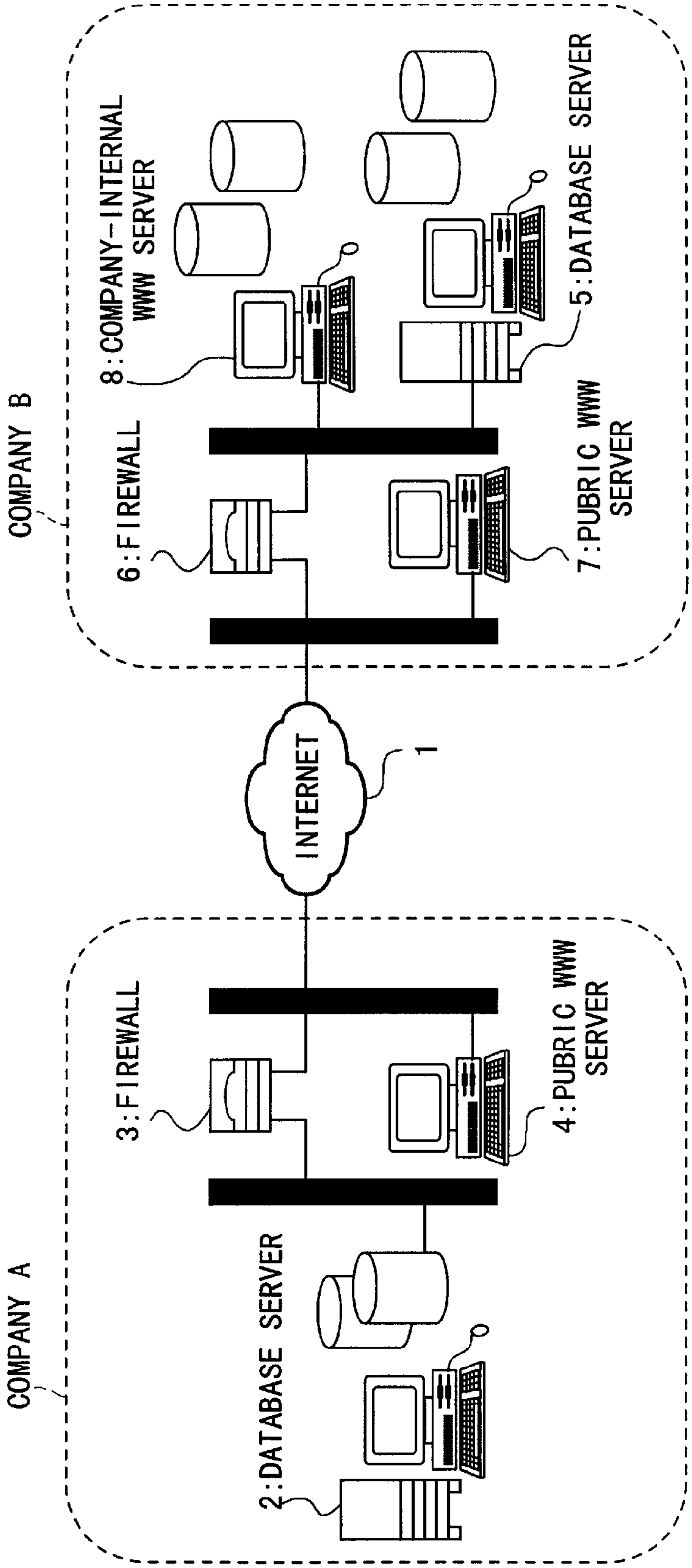
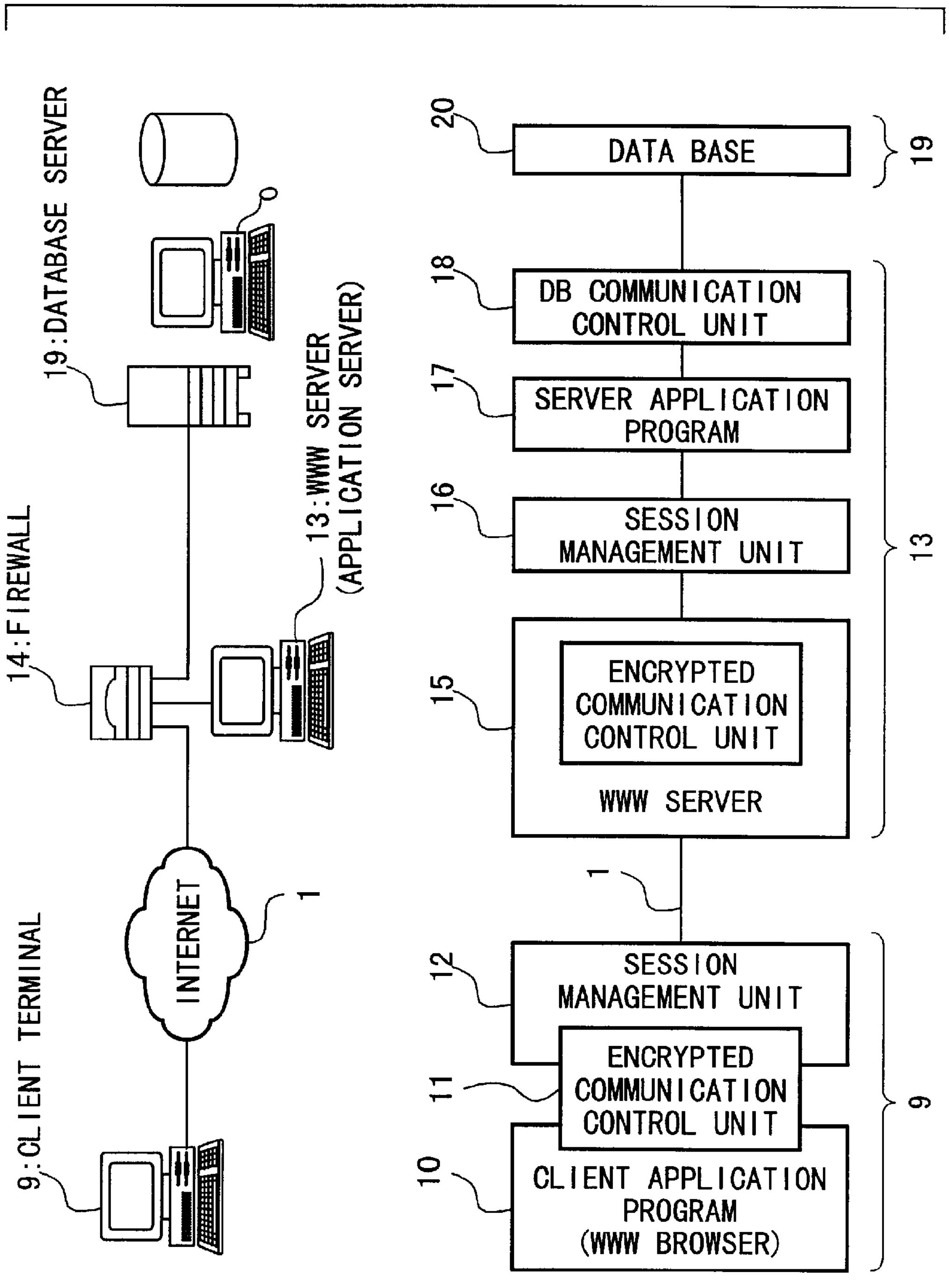

COMPANY A
2:DATABASE SERVER
3:FIREWALL
4:PUBRIC WWW SERVER
INTERNET
1
COMPANY B
6:FIREWALL
7:PUBRIC WWW SERVER
8:COMPANY-INTERNAL WWW SERVER
5:DATABASE SERVER

FIG.6


9:CLIENT TERMINAL
INTERNET
1
14:FIREWALL
13:WWW SERVER
(APPLICATION SERVER)
19:DATABASE SERVER
10
CLIENT APPLICATION
PROGRAM
(WWW BROWSER)
11
ENCRYPTED
COMMUNICATION
CONTROL UNIT
12
SESSION
MANAGEMENT UNIT
9
1
15
ENCRYPTED
COMMUNICATION
CONTROL UNIT
WWW SERVER
16
SESSION
MANAGEMENT UNIT
17
SERVER APPLICATION
PROGRAM
18
DB COMMUNICATION
CONTROL UNIT
13
20
DATA BASE
19

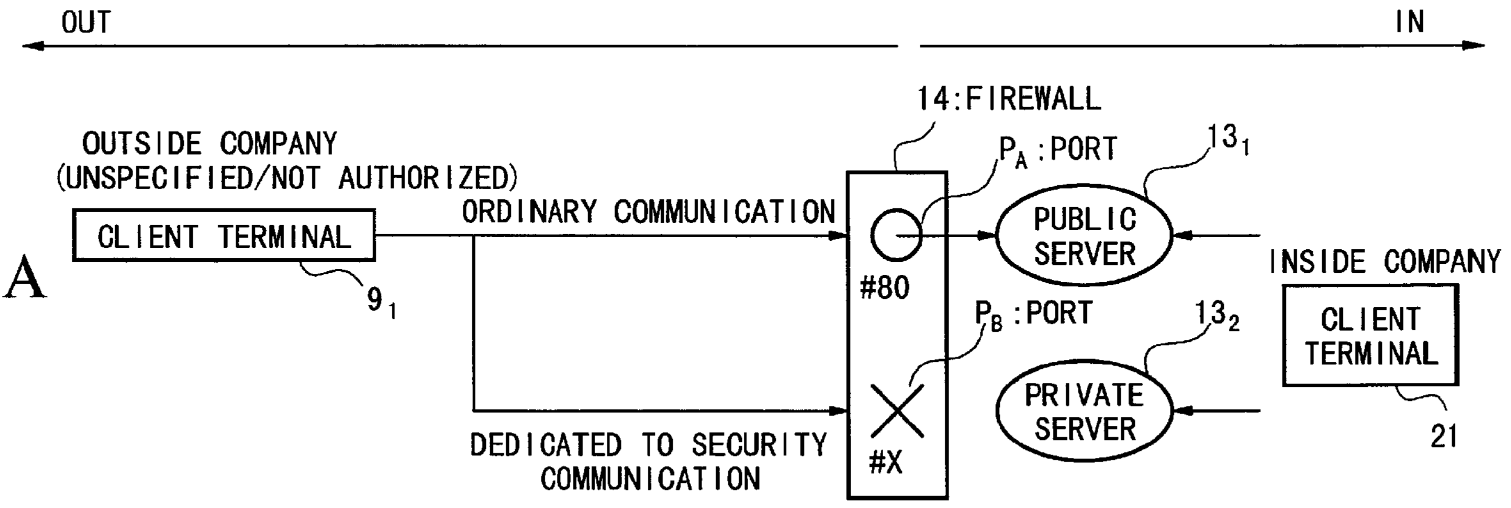

OUT
IN
14:FIREWALL
OUTSIDE COMPANY
(UNSPECIFIED/NOT AUTHORIZED)
CLIENT TERMINAL
9_1
ORDINARY COMMUNICATION
DEDICATED TO SECURITY COMMUNICATION
#80
#X
P_A :PORT
P_B :PORT
PUBLIC SERVER
13_1
PRIVATE SERVER
13_2
INSIDE COMPANY
CLIENT TERMINAL
21

FIG.7A

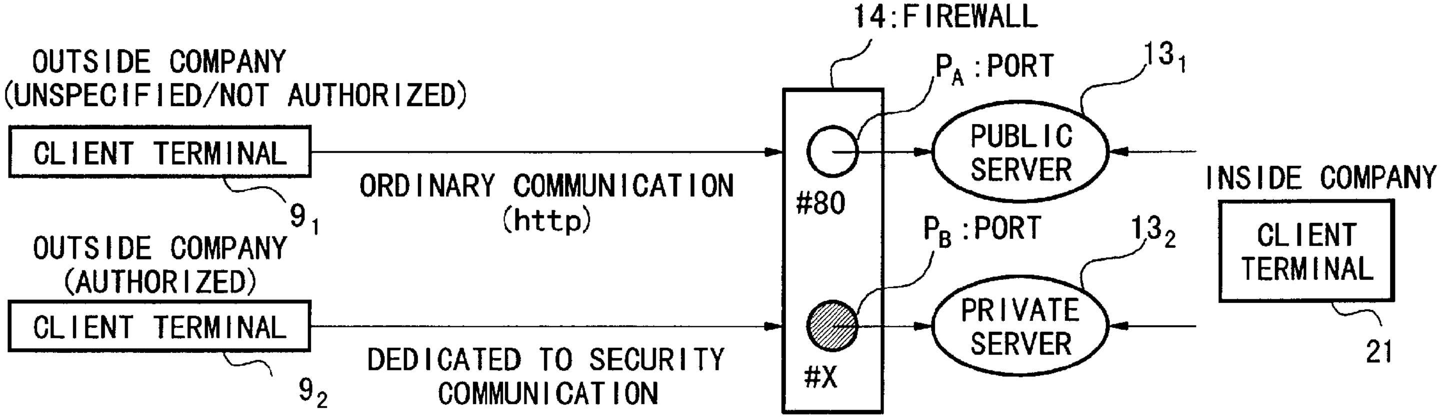
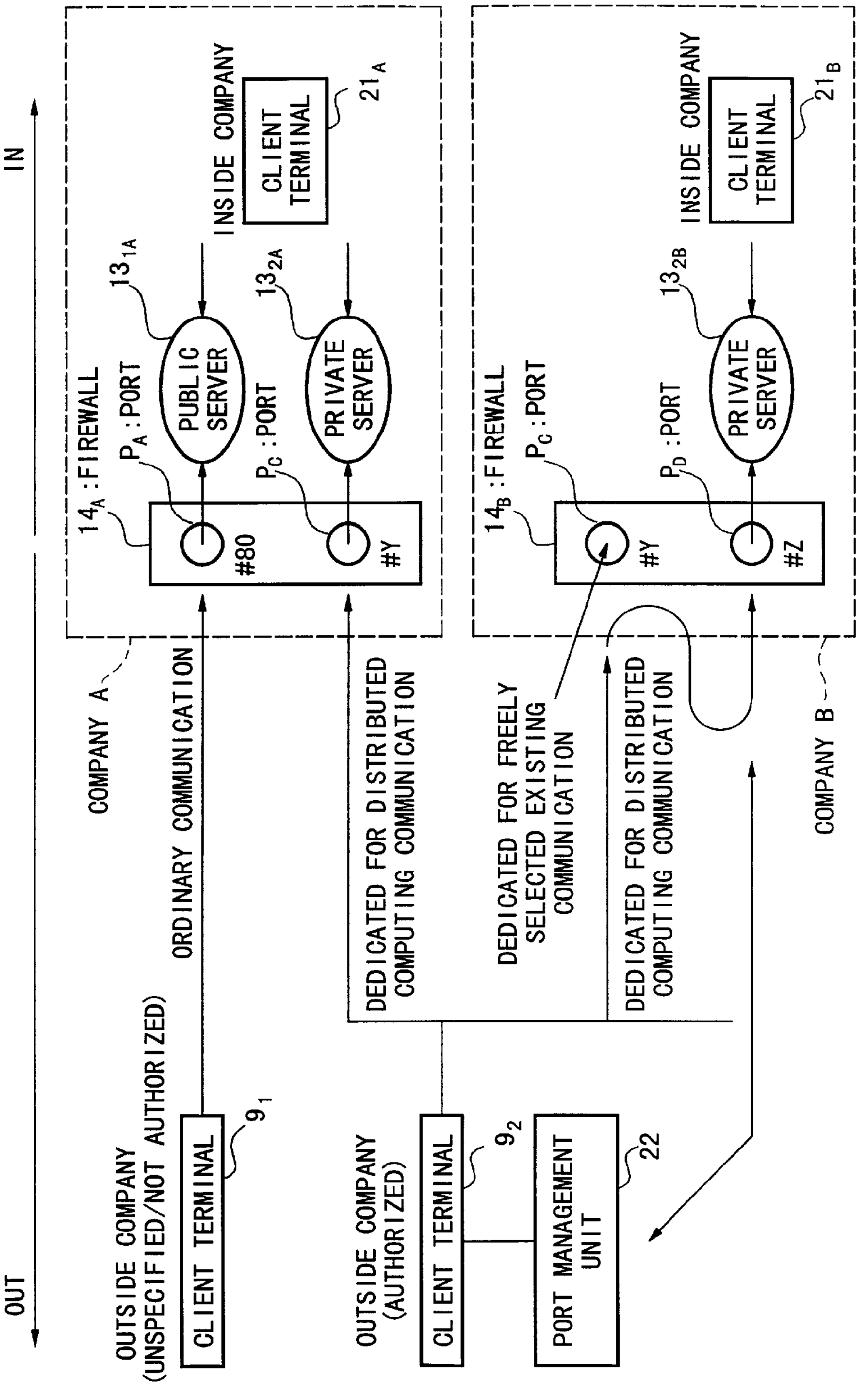

14:FIREWALL
OUTSIDE COMPANY
(UNSPECIFIED/NOT AUTHORIZED)
CLIENT TERMINAL
9_1
OUTSIDE COMPANY
(AUTHORIZED)
CLIENT TERMINAL
9_2
ORDINARY COMMUNICATION
(http)
DEDICATED TO SECURITY COMMUNICATION
#80
#X
P_A :PORT
P_B :PORT
PUBLIC SERVER
13_1
PRIVATE SERVER
13_2
INSIDE COMPANY
CLIENT TERMINAL
21

FIG.7B

FIG.8


OUT
IN
OUTSIDE COMPANY (UNSPECIFIED/NOT AUTHORIZED)
CLIENT TERMINAL
9₁
ORDINARY COMMUNICATION
OUTSIDE COMPANY (AUTHORIZED)
CLIENT TERMINAL
9₂
PORT MANAGEMENT UNIT
22
DEDICATED FOR DISTRIBUTED COMPUTING COMMUNICATION
DEDICATED FOR FREELY SELECTED EXISTING COMMUNICATION
DEDICATED FOR DISTRIBUTED COMPUTING COMMUNICATION
COMPANY A
COMPANY B
14A : FIREWALL
PA : PORT
#80
PUBLIC SERVER
13 1A
PC : PORT
#Y
PRIVATE SERVER
13 2A
INSIDE COMPANY
CLIENT TERMINAL
21A
14B : FIREWALL
PC : PORT
#Y
PD : PORT
#Z
PRIVATE SERVER
13 2B
INSIDE COMPANY
CLIENT TERMINAL
21B

FIREWALL PORT SWITCHING

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a network system which is employed in the access to servers via networks from client terminals.

This application is based on patent application No. Hei 10-146372 filed in Japan, the contents of which are incorporated herein by reference.

2. Description of the Related Art

Conventionally, in LAN (local area network) environments in corporations, various types of controls necessary for the main business were employed, so that the connection of the LAN system or the like via the internet has been difficult as a result of problems regarding the advisability of protocols for passage through firewalls to be described hereinbelow, and the like.

However, recently, as a result of the penetration of distributed computing technologies and the spread of Java, it has become possible to construct network systems by means of connecting company-wide LAN systems via the internet. Here, when this type of network system is constructed, by means of installing a firewall, security is maintained.

Here, a firewall is a system which is installed at the point of attachment between the information system itself and the internet, and which serves the function of a firewall; it prevents the unpermitted intrusions from unauthorized individuals and keeps out computer viruses.

Furthermore, in network systems having firewalls such as that described above, there may be limitations in accordance with security policies with respect to classifications of protocols which may be employed in this environment, and thereby, by disallowing the passage of freely selected protocols, security is maintained.

FIG. **5** shows the outlines of the composition of the conventional network system described above. In this figure, reference **1** indicates the internet, in which a plurality of networks are connected to one another, and in the example shown in FIG. **5**, internet **1** connects the LAN of company A and the LAN of company B. In company A, reference **2** indicates a database server which stores various databases in a storage unit, and this is connected to internet 1 via firewall **3**.

It is only possible for authorized terminals to access the database server **2** via firewall **3**. Unauthorized terminals are incapable of accessing database server **2** through firewall **3**. Reference **4** indicates a public WWW (world wide web) server which is connected to the internet **1**, and this is freely accessible by any terminal irrespective of its authorized or non-authorized status.

In company B, reference **5** indicates a database server which stores various databases in the storage unit thereof; this is connected to internet **1** via firewall **6**. Only authorized terminals are capable of accessing this database server **5** via firewall **6**. Reference **7** indicates a public WWW server which is connected to internet 1, and this server is accessible by terminals irrespective of their authorized or non authorized status. Reference **8** indicates a company internal WWW server which is connected to internet **1** via firewall **6**; this company internal WWW server **8** may be accessed via firewall **6** only by authorized terminals.

FIG. **6** shows the main parts of the composition of a conventional network server. In this figure, reference **9** indicates a client terminal which is installed on the client side and is connected to internet **1**. This client terminal **9** conducts access to the WWW server **13** and the database server **19** described hereinbelow via internet **1**. In client terminal **9**, reference **10** indicates a client application program which is executed by client terminal **9**; this program serves to conduct communication control, encryption control, protocol control, and the like. Furthermore, the client application program **10** is a program which is executed when other company-side applications are employed from client terminal **9** via internet **1**. Reference **11** indicates an encrypted communication control unit, which has the function of controlling an encoding dedicated protocol for conducting encryption and decoding of data grams passing through specified protocol service ports defined in advance, irrespective of the attributes of the data (for example, an SSL or secure socket layer). Reference **12** indicates a session management unit which manages the sessions.

WWW server **13** is connected to internet **1** via firewall **14**, and is a terminal which functions using the startup from client terminal **9** as an opportunity. Here, a plurality of ports are provided in firewall **14**, and these ports may be broadly classified into standard ports for the communication of protocols from unauthorized client terminals **9**, and security communication ports for communicating only those protocols from authorized client terminals **9**.

In the WWW server **13** described above, reference **15** indicates an encrypted communication control unit having a function identical to that of the encrypted communication control unit **11** described above. Reference **16** indicates a session management unit which manages the sessions. Reference **17** indicates a server application program which is executed by WWW server **13**, and which is employed in the control of communications with client terminals **9**. Reference **18** indicates a DB (database) communication control unit which conducts the control of access to database **20** described hereinbelow. Database server **19** stores database **20** in the memory unit thereof.

Here, the operation of the network system shown in FIG. **6** will be explained using the operations explanatory diagrams shown in FIGS. **7**A and **7**B. FIG. **7**A serves to explain the access operation from unauthorized company external client terminals 9_1, while FIG. **7**B serves to explain the access operation from unauthorized and authorized client terminals 9_1 and 9_2.

Here, in FIGS. **7**A and **7**B, client terminal 9_1 corresponds to one unauthorized client terminal **9** in FIG. **6**, and is located outside the company. Client terminal 9_2 corresponds to a different authorized client terminal **9** in FIG. **6**, and is also located outside the company.

The firewalls **14** shown in FIGS. **7**A and **7**B have ports P_A and P_B, and ports P_A are ports which are assigned the port number **#80**, and which are installed for the purposes of access from an unspecified large number of client terminals. Accordingly, the port number **#80** of port P_A described above is public. On the other hand, port P_B is provided with a port number #X, and is installed for the purposes of access from authorized client terminals 9_2. Accordingly, this port number #X of ports P_B is a number which may only be employed in communications by the clients of client terminals 9_2 which have authorization. In other words, access to ports P_B is only possible from specified client terminals 9_2.

The public server 13_1 and private server 13_2 shown in FIGS. **7**A and **7**B correspond to the WWW server **13** shown in FIG. **6**. Here, a client terminal 9_1 is provided with access to public server 13_1 via internet **1** and port P_A of firewall **14**. On the other hand, a client terminal 9_2 accesses private server 13_2 via internet **1** and the port P_B of firewall **14**. Reference **21** indicates a client terminal located within the company; since security is maintained on the inside of the firewall, this terminal may directly access public server 13_1 and private server 13_2.

In FIG. **7**A, the unauthorized client terminal 9_1 commonly accesses public server 13_1 through port P_A of firewall **14** using http (hypertext transfer protocol). At this time, the http described above is capable of passing through port P_A.

Here, when an attempt is made to access private server 13_2 from client terminal 9_1, since the client of client terminal 9_1 does not know the port number #X of port P_B, it is impossible to pass through the firewall **14**. In other words, the http from client terminal 9_1 is not capable of passing through port P_B, so that no communication is established between client terminal 9_1 and private server 13_2. Accordingly, in this case, client terminal 9_1 is incapable of accessing private server 13_2, and security is maintained.

On the other hand, in FIG. **7**B, in the case in which client terminal 9_2 attempts to access private server 13_2, client terminal 9_2 employs the security communication dedicated protocol, and first accesses port P_B. At this time, the protocol described above is capable of passing through port P_B, so that client terminal 9_2 is capable of accessing private server 13_2.

In the conventional network system described above, more secure communication between companies are realized using a firewall; however, the needs are great.

However, in the firewall environment of the network system described above, the structure is one in which firewalls having a distributed structure are distributed stepwise in a plurality of steps, so that this presents a problem in that in order to enable a passage of a single new protocol through the firewall, an enormous amount of preparation and work are required. Examples of this preparation and work include the resetting of the firewall ports shown in FIG. **6**, and the modification of the use of the client application program **10** and the server application program **17**.

Here, the problems of the conventional network system will be explained with reference FIG. **8**.

In FIG. **8**, in the parts corresponding to FIGS. **7**A and **7**B, the same reference numbers are employed. In company A shown in FIG. **8**, reference 14_A indicates a firewall having a function identical to that of the firewall **14** shown in FIG. **7**; this is provided between internet **1** (see FIG. **6**) and public server 13_{1A} and private server 13_{2A}. Here, firewall 14_A is provided with ports P_A and PC.

The port P_A described above is given the port number #**80**, and is a port which is provided for the purposes of access from an unspecified large number of client terminals. On the other hand, port PC is given the port number #Y, and is provided for the purposes of access from authorized client terminals 9_2 (distributed computing communications). This port PC is the security dedicated port. Accordingly, the port number #Y of PC may be employed in communications only by the clients of the authorized client terminal 9_2. In other words, only specified client terminals 9_2 are capable of accessing port PC. Reference 21_A indicates a client terminal which is installed in company A, which accesses public server 13_{1A} and private server 13_{2A}.

Furthermore, in company B, reference 14_B indicates a firewall which is provided between internet **1** and private server 13_{2B}, and this is also provided with port PD and port PC which is dedicated to distributing computing communication. Port PC described above is provided with port number #Y, while port PD is provided with a port number #Z. The port number #Y of port PC makes possible communications only from clients of the authorized client terminal 9_2. The ports PC and PD are security dedicated ports.

In the structure described above, unauthorized client terminal 9_1 commonly accesses public server 13_{1A} via the port P_A of firewall 14_A using http (hyper text transfer protocol). At this time, this http is capable of passing through port P_A. Client terminal 9_1 is incapable of accessing the servers through port PC of firewall 14_A and ports PC and PD of firewall 14_B in the same way as in the operations described above.

On the other hand, when access is conducted from client terminal 9_2 to private server 13_{2A}, client terminal 9_2 first accesses port PC of firewall 14_A using a security communications dedicated protocol. At this time, this protocol is capable of passing port PC, so that client terminal 9_2 is capable of accessing private server 13_{2A}.

Here, the case is explained in which the client terminal 9_2 accesses the private server 13_{2B} via port PC of firewall 14_B, in the state in which the firewall 14_B has already been allocated for other service protocols.

In this case, port PC is closed, so that it is necessary to establish port PD in firewall 14_B. The information regarding this modification of the port setting must be communicated to the manager of client terminal 9_2.

Here, a port management unit **22**, which manages the port data in the plurality of firewalls, is provided in client terminal 9_2.

Here, in the conventional network system (see FIG. **8**), in order to realize distributed computing, access should be made possible from client terminal 9_2 and the like to all destination systems (systems within other companies) in which all necessary functions (server applications) are present, and security control for satisfying all security policies is conducted.

However, as explained with reference to FIG. **8**, in conventional network systems, the port setting modification rules differ from company to company, and this increases the complexity of the management of definitional data to port managing unit **22**, and makes the control more complex.

Accordingly, in order to add conditions for port setting of this type, and to conduct the execution of applications with respect to work units, it is necessary to research and develop extremely complex installation methods. In particular, with respect to changes in installation with respect to security matters, this is a necessary and extremely serious matter for consideration for the company units, and represents an obstacle to the rapid realization of such systems.

SUMMARY OF THE INVENTION

For this reason, it is an object of the present invention to provide a network system which does not require individual security dedicated ports for the establishment of firewall security.

In this invention, the network system is provided with authorized client terminals connected to the network, with a server connected to the network, and with a firewall which is interposed between the server and the network. The object described above may be obtained by means of a server for a network system which, in the case in which, when a client terminal accesses the server by means of a public protocol via a port with a publicly known port number in a firewall, the accessing client terminal is an authorized terminal, downloads to the client terminal, via the port with the publicly known port number, a program for realizing effective dedicated control solely between the client terminal and server, conducts data communication with the client terminal via the network and the port with the publicly known port number by means of the dedicated control.

In the present invention, by means of using dedicated control, the port in the firewall is constantly a port with a known port number. Accordingly, port management on the client terminal side is unnecessary. By means of this, it is possible to obtain a network system which does not require independent security dedicated ports to establish firewall security.

Furthermore, in the network system, in the case in which there is a proxy server which conducts the port switching in the firewall, the network server communicates the first port to the client terminal as the communication port, and sets the port it itself employs as a second port having a port number other than the publicly known port number. Additionally, the network server conducts data communication with the client terminal via the networks, the firewall, and the proxy server using the dedicated protocol.

By means of this, even in the case in which a proxy server is installed in the network system, it is possible to obtain a network system which does not require an independent security dedicated port in order to establish firewall security.

The network system server preferably conducts the encryption and decoding of data in the data communication.

By means of this, an effect is obtained whereby secure communications are realized.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows the structure of the main parts of a network system in accordance with an embodiment of the present invention.

FIG. **2** shows the structure in the case in which a network system in accordance with this embodiment is applied to distributed communications between companies.

FIG. **3** serves to explain the operation of the network system of this embodiment.

FIG. **4** serves to explain the structure and operation of the network system of another embodiment.

FIG. **5** shows the outline of the structure of a conventional network system.

FIG. **6** shows the main parts of the structure of a conventional network system.

FIGS. **7**A and **7**B serve to explain the operation of the conventional network system.

FIG. **8** serves to explain the problems present in the conventional network system.

DESCRIPTION OF PREFERRED EMBODIMENTS

The embodiments described hereinbelow should not be construed as limiting the invention described in the Claims. Furthermore, it is not the case that all combinations of features described in the embodiments are necessarily required in order to achieve the object of the present invention.

Hereinbelow, embodiments of the present invention will be explained with reference to the figures. FIG. **1** shows the main parts of the structure in a network system in accordance with an embodiment of the present invention. In this figure, parts corresponding to the parts in FIG. **6** are given identical reference numbers.

Reference **30** shown in FIG. **1** indicates a client terminal which is located at the client side; this is connected to internet **1**. This client terminal **30** accesses WWW server **34** and database server **19** described hereinbelow via internet **1**.

In client terminal **30**, reference **31** indicates a client application program which is executed by client terminal **30**; this program serves to conduct communication control, encryption control, and protocol control and the like. Furthermore, client application program **10** is executed when the dedicated control described hereinbelow is employed, and is a program which is executed when the other company-side applications are used from client terminal **30** via internet **1** and firewall **35**.

Reference **32** indicates a distributed computing communication control unit; this dynamically connects a portion of a server application program **38** to be described hereinbelow with the client application program **31**, and copies this. Furthermore, distributed computing communication control unit **32** has a communication protocol function which serves to realize and execution environment such that a portion of the server application program **38** described above may be manipulated just as if it were a preexisting work application program in client terminal **30**. Reference **33** indicates an encrypted communication control unit, which has the same function as the encrypted communication control unit **11** in FIG. **6**.

WWW server **34** is connected to internet **1** via firewall **35**, and is a terminal which functions using the startup from client terminal **30** as an opportunity. A plurality of ports are provided in firewall **35**; however, client terminal **30** employs the port with port number #**80** in communications between client terminal **30** and WWW server **34** irrespective of whether client terminal **30** is authorized or not. This port with the port number #**80** is public, and serves to allow the passage of common protocols such as http or the like. The setting state of the ports of this firewall **35**, and the details of the communication protocols between client terminal **30** and WWW server **34** will be discussed hereinbelow.

In WWW server **34**, reference **36** indicates an encrypted communication control unit which has a function similar to that of the encrypted communication control unit **11** described above. Reference **37** indicates a distributed computing communication control unit which has a similar function to that of the distributed computing communication control unit **32**. Reference **38** indicates a server application program which is executed by WWW server **34** and which is used in the control of communications with client terminal **30** and the like. Furthermore, server application program **38** is a program which is executed when dedicated protocols described hereinbelow are employed. Reference **39** indicates a DB (database) communication control unit which conducts access control with respect to database **20**.

Here, the structure resulting when the network system shown in FIG. **1** is applied to distributed computing communications between companies will be explained with reference to FIG. **2**. In the network system shown in FIG. **2**, the system within a company A and the system within a company B are connected via internet 1 (see FIG. **1**), and a non-authorized client terminal $\mathbf{30}_1$, and an authorized client terminal $\mathbf{30}_2$ are connected to internet **1**.

Here, the client terminal $\mathbf{30}_1$ shown in FIG. **2** corresponds to one nonauthorized client terminal **30** shown in FIG. **1** and is located outside the companies. Client terminal $\mathbf{30}_2$ corresponds to another client terminal **30** shown in FIG. **1** which is authorized and is also located outside the companies. Here, the client application program **31** shown in FIG. **1** is stored in the storage units of client terminals $\mathbf{30}_1$ and $\mathbf{30}_2$.

In company A, firewall **35**$_A$ corresponds to firewall **35** in FIG. **1**, and has a port P_A. This port P_A is given a port number of **#80**, and this port is set for the access of an unspecified large number of client terminals. In actuality, a plurality of logic ports are provided in firewall **35**$_A$, and the port numbers of these ports may be freely set. However, in the explanation which follows, the only port which is employed is that which has the port number **#80**.

Reference **34**$_{1A}$ indicates a public server corresponding to the WWW server **34** shown in FIG. **1**, and this is accessed by client terminal **30**$_1$ via internet **1** and firewall **35**$_A$. Reference **34**$_{2A}$ indicates a private server corresponding to the WWW server shown in FIG. **1**, and this is accessed by an authorized client terminal **30**$_2$ via internet **1** and firewall **35**$_A$ (port P_A) by means of dedicated protocols described hereinbelow. Here, the server application program **38** shown in FIG. **1** is stored in the storage unit of public server **34**$_{1A}$ and private server **34**$_{2B}$. Client terminal **21**$_A$ is located within company A, and accesses both public server **34**$_{1A}$ and private server **34**$_{2A}$.

On the other hand, in company B, firewall **35**$_B$ corresponds to the firewall **35** shown in FIG. **1**, and has a port P_A. This port P_A is located with port number #80. The function of this firewall **35**$_B$ is the same as the function of the firewall **35**$_A$. Reference **34**$_{2B}$ indicates a private server which may be accessed by client terminal **30**$_2$ via internet **1** and firewall **35**$_B$ (port P_A) using dedicated protocols. The server application program **38** shown in FIG. **1** is provided in the storage unit of this private server **34**$_{2B}$. The client terminal **21**$_B$ is located within company B, and accesses private server **34**$_{2B}$.

Next, the operation of the network system of the embodiment described above will be explained with reference to FIG. **3**. In this figure, the parts corresponding to the parts of FIG. **2** are given identical reference numbers, and a description thereof will be omitted here. In the figure, the example is depicted in which the authorized client terminal **30**$_2$ shown in FIG. **2** accesses the private server **34**$_{2A}$ via internet **1** and the firewall **35**$_A$ within company A.

Furthermore, in the firewall **35**$_A$ shown in FIG. **3**, port P_A, and a port p_A differing from this port P_A are provided; however, the port number **#80** is provided to all of these ports in a timely manner. Furthermore, in firewall **35**$_A$, the ports to which the port number **#80** is assigned are altered.

Furthermore, the client application program (AP) **42** shown in FIG. **3** corresponds to the client application program **31** shown in FIG. **1**, and has a driver (RApp_C) **43**. This driver **43** comprises a portion of the function realized by client application program **42**, and serves to control the control sequence between client terminal **30**$_2$ and private server **34**$_{2A}$.

Furthermore, dedicated server processing program (Regist) **44** is a program forming a portion of server application program **38**, and serves to conduct the control of the communications between private server **34**$_{2A}$ and client terminal **30**$_2$. This dedicated server processing program **44** comprises a remote object program (stub) **45** and a communication control program (Skel) **46**.

This remote object program **45** is transmitted to client terminal **30**$_2$ via firewall **35**$_A$ and internet **1**, and is then executed by client terminal **30**$_2$; it serves to control communications. On the other hand, the communication control program **46** is executed by private server **34**$_{2A}$, and forms a pair with the remote object program **45**, serving to control communications.

In the structure described above, when the private server **34**$_{2A}$ is started up, the dedicated server processing program **44** is executed, and private server **34**$_{2A}$ enters a state in which operations are possible. In this state, in step SA1, http **40** and client authorization data indicating authorization are sent from client terminal **30**$_2$ to firewall **35**$_A$ via internet **1**. Now, if it is assumed that the port P_A of firewall **35**$_A$ has been assigned the port number **#80**, then http **40** passes through port P_A of firewall **35**$_A$, and enters private server **34**$_{2A}$.

By means of this, private server **34**$_{2A}$ makes a determination as to whether the client terminal **30**$_2$ is authorized or not, from the client authorization data included in a portion of the communication data initially transmitted, and if the server-side authorization fails, the server does not conduct further operations.

In the present case, since client terminal **30**$_2$ is an authorized terminal, private server **34**$_{2A}$ recognizes client application program **42** by means of an httpd (http daemon) **41**.

Then, in step SA2, private server **34**$_{2A}$ downloads the client application program **42** in the form of a Java applet to client terminal **30**$_2$ via port P_A and internet **1**. By means of this, in client terminal **30**$_2$, the client application program **42** is executed, and thereby, distributed computing communications are initiated.

Next, in step SA3, client terminal **30**$_2$ submits a request to private server **34**$_{2A}$ for data relating to the ports (numbers) in firewall **35**$_A$ used in distributed computing communications, using driver **43** and via internet **1** and port P_A. By means of this, in step SA4, private server **34**$_{2A}$ reserves a port p_A in place of port P_A, and the port number **#80** is assigned to port p_A. That is to say, by means of this reservation, the port having the number **#80** is changed from port P_A to port p_A. The subsequent protocol sequence is all conducted via port p_A (with the port number **#80**).

Next, in step SA4, private server **34**$_{2A}$ transmits data relating to port p_A (port number **#80**), which was reserved as the port for conducting the protocol sequence, to client terminal **30**$_2$ via port p_A and internet 1. By means of this, client terminal **30**$_2$ recognizes that the port of firewall **35**$_A$ which is to be subsequently used is port p_A (port number **#80**).

Next, in step SA5, client terminal **30**$_2$ sends data serving to request a download of the remote object program **45**, which is necessary for communications via the stipulated port p_A (port number **#80**), to private server **34**$_{2A}$ via internet **1** and port p_A. By means of this, in step SA6, private server **34**$_{2A}$ downloads remote object program **45** to client terminal **30**$_2$ via port p_A and internet **1**.

By means of this, in client terminal **30**$_2$, the remote object program **45** is executed. Thereinafter, in the manner shown in steps SA7–SA10, data communication is conducted between client terminal **30**$_2$ and private server **34**$_{2A}$ via internet **1** and the port p_A of firewall **35**$_A$. Furthermore, in this data communication, the encryption and decoding of the data is conducted by the encrypted communication control units **33** and **36** shown in FIG. **1**, so that secure communications may be realized.

As described above, in accordance with the network system of the present embodiment described above, by means of employing a dedicated protocol, a structure is achieved in which the port in firewall **35**$_A$ is always assigned the port number **#80**, so that port management in client terminal **30**$_2$ is unnecessary. From this, in accordance with the network system of the embodiment described above, an effect is achieved whereby it is possible to obtain a network system which does not require independent security dedicated ports for the setting of firewall security.

Furthermore, in accordance with the network system of the embodiment described above, an effect is achieved whereby it is possible to safely use distributed computing communications, the passage of which has not yet been recognized, without conducting special setting modification of the existing internet security policies.

Furthermore, in accordance with the network system of the embodiment described above, an effect is achieved whereby it is possible to realize practical convenience by means of conducting encryption and decoding with protocol levels with respect to all data passing through the ports of firewall **35**$_A$.

Furthermore, in accordance with the embodiment described above, since it is not necessary to add large modifications to the internet design established in the various companies, it is possible to complete design and installation in connected companies in an extremely short period of time, and accordingly, it is possible to construct a distributed system for connecting these companies in a short period of time.

In the foregoing, a network system in according with an embodiment of the present invention was described in detail; however, the actual structure is not limited to this embodiment, and design modifications and the like are also included in the present invention to the extent that they do not depart from the essential idea of the present invention.

For example, in the network system in accordance with the embodiment described above, the structure shown in FIG. **3** was explained; however, it is also possible to adopt the structure shown in FIG. **4** in place thereof.

Hereinbelow, the network system shown in FIG. **4** will be explained.

In FIG. **4**, the parts corresponding to the parts in FIG. **3** are given identical reference numbers, and an explanation thereof will be omitted here. In FIG. **4**, a proxy server **47** is also provided. Furthermore, in FIG. **4**, firewall **35**$_A$ is provided with a port P_A having a port number #**80**, and a port P_B having a port number differing from that of port P_A. The port number of port P_B may be set to, for example, #X.

Proxy server **47** is provided at firewall **35**$_A$ (or private server **34**$_{2A}$), and this server serves to allow the passage of data from the private network on the company A side to a public network such as internet **1** or the like, or in the opposite direction.

In FIG. **4**, proxy server **47** outputs data inputted into port P_A to private server **34**$_{2A}$ via port P_B, and also outputs data inputted into port P_B to internet 1 via port P_A, thus having a port-switching function. In other words, by means of proxy server **47**, in the case in which private server **34**$_{2A}$ is seen from client terminal **30**$_2$, the accessible port is set to port P_A, while in the case in which the private server **34**$_{2A}$ is seen from private server **34**$_{2A}$, the accessible port is set to port P_B.

In the structure described above, when private server **34**$_{2A}$ is started up, the dedicated sever processing program **44** is executed, and private server **34**$_{2A}$ enters a state in which operations are possible. In this state, in step SB**1**, http **40** is outputted from client terminal **30**$_2$ to firewall **35**$_A$ via internet **1**. Now, if it is assumed that port number #**80** has been assigned to port P_A of firewall **35**$_A$, then http **40** passes through port P_A of firewall **35**$_A$ and enters private server **34**$_{2A}$. By means of this, private server **34**$_{2A}$, in the manner of the operations described above, recognizes client application program **42** by means of httpd **41**.

Next, in step SB2, private server **34**$_{2A}$ downloads client application program **42** to client terminal **30**$_2$ as a Java applet or the like via port P_A and internet **1**. By means of this, the client application program **42** is executed in client terminal **30**$_2$, and by means thereof, distributed computing communications are initiated.

Next, in step SB**3**, client terminal **30**$_2$ sends a request via internet **1** and port P_A to private server **34**$_{2A}$ for data relating to the port (number) in firewall **35**$_A$, which is used in distributed computing communications, using driver **43**. By means of this, in step SB**4**, private server **34**$_{2A}$ reserves port P_B (port number #X) as the port which it itself uses, and sends data relating to the port P_B (port number #X) to proxy server **47**. Furthermore, private server **34**$_{2A}$ sends data relating to port P_A (port number #**80**) , the port used by client terminal **30**$_2$, to client terminal **30**$_2$ via port P_A and internet **1**.

By means of this, in proxy server **47**, the port switching of port P_A->port P_B (port P_A<-port P_B) is defined.

By means of this definition, the protocol sequence in the case in which client terminal **30**$_2$ is viewed from private server **34**$_{2A}$ is conducted in port P_B of firewall **35**$_A$, while the protocol sequence in the case in which private server **34**$_{2A}$ is viewed from client terminal **30**$_2$ is conducted in port P_A of firewall **35**$_A$. That is to say, the private server **34**$_{2A}$ recognizes the port P_B (port number #X) as the port for conducting distributed computing communications, while client terminal **30**$_2$ recognized port P_A (port number #**80**) as this port.

Next, in step SB**5**, client terminal **30**$_2$ sends data for the purpose of requesting download of the remote object program **45**, which is necessary for communications via the stipulated port P_A (port number #**80**), to internet **1**. By means of this, the data switching (port P_A->port P_B) is conducted in proxy server **42**, and the data described above are inputted into private server **34**$_{2A}$ via ports P_A and P_B.

By means of this, in step SB **6**, private server **34**$_{2A}$ outputs remote object program **45** via port P_B. At this time, by means of proxy server **47**, the port switching (port P_B->port P_A) is conducted, and remote object program **45** is downloaded to client terminal **30**$_2$ via port P_B, port P_A, and internet **1**.

By means of this, in client terminal **30**$_2$, remote object program **45** is executed. Thereinafter, as shown in steps SB**7**–SB**10**, data communication is conducted between client terminal **30**$_2$ and private server **34**$_{2A}$ via internet **1** and ports P_A and P_B of firewall **35**$_A$. Furthermore, in this data communication, the encryption and decoding of the data are conducted by means of the encrypted communication control units **33** and **36** shown in FIG. **1**, so that secure communication can be realized.

Furthermore, in the network system of the embodiment described above, the computer readable program for executing the functions described above may be stored in a computer-usable medium, and this program stored in this medium may be read out to the computer system and executed. What is meant here by a computer system includes hardware such as an OS (operating system) or peripheral devices. Furthermore, the computer system may include a home page providing environment (or display environment) if a WWW system is employed.

As explained above, by means of the present invention, by using dedicated protocols, the port in the firewall is constantly set to a port having a publicly known port number, so that port management is not required on the client terminal side.

From this, by means of the present invention, an effect is obtained whereby it is possible to obtain a network system which does not require independent security dedicated ports for setting the firewall security.

Furthermore, by means of adding an encrypted communication control unit, the encryption and decoding of the data can be conducted, so that an effect is provided whereby it is possible to realize secure communications.

What is claimed is:

1. A network system comprising an authorized client terminal connected to a network, a server connected to a network, and a firewall interposed between said server and said network, wherein:

said client terminal accesses said server using publicly known protocol via a port with a publicly known port number in said firewall;

in the case when said client terminal conducting access is authorized, said server downloads a program for realizing effective dedicated protocol solely between said client terminal and said server to said client terminal via said port having said publicly known port number; and

said client terminal and said server conduct data communication via said network and said port having said publicly known port number using said dedicated protocol by executing said program,

wherein said network system further comprises a proxy server which conducts port switching in said firewall, and wherein:

in the case when said client terminal conducting access is authorized, said server downloads a program for realizing effective dedicated protocol solely between said client terminal and said server to said client terminal via a first port having a publicly known port number, whereafter communicates said first port to said client terminal as a port for communications, and sets a port used by said server as a second port having a port number other than said publicly known port number;

said proxy server switches a port seen from said client terminal from said first port to said second port, and switches a port seen from said server from said second port to said first port; and

said client terminal and said server conducts data communication via said network, said firewall, and said proxy server using said dedicated protocol by executing said program.

2. A network system, according to claim 1, wherein:

said client terminal is provided with a first encrypted communication control unit which conducts encryption and decoding of data in said data communication; and

said server is provided with a second encrypted communication control unit which conducts encryption and decoding of data in said data communication.

3. A server for a network system, wherein is provided a processing unit which, in the case when a client terminal conducting access by publicly known protocol, via a port having a publicly known port number in a firewall, is authorized, downloads a program, for realizing effective dedicated protocol solely between said client terminal and said server, to said client terminal via said port having said publicly known port number, and said server conducts data communication with said client terminal via said network and said port having said publicly known port number conducts using said dedicated protocol,

wherein, in the case when a proxy server which conducts port switching in said firewall is present in said network system,

said network server is further provided with a processing unit which communicates a first port to said client terminal as a port for communication, and sets a port used by said server as a second port having a port number other than said publicly known port number, and

the server conducts data communication with said client terminal via said network, said firewall, and said proxy server using said dedicated protocol.

4. A server for network system, according to claim 3, wherein said network system server is further provided with an encrypted communication control unit which conducts encryption and decoding of data in said data communication.

5. A computer program product containing a computer readable program recorded on a computer usable medium, said program affecting the process of:

determining as to whether a client terminal conducting access using publicly known protocol via a port having a publicly known port number in a firewall is authorized or not;

in the case when the client terminal is authorized, downloading a program, for realizing effective dedicated protocol solely between said client terminal and a server, to said client terminal via said port having said publicly known port number; and

conducting data communication with said client terminal using said dedicated protocol via said network and said port having said publicly known port number,

wherein, in the case when a proxy server which conducts port switching in said firewall is present in said network system, said program further effecting the process of:

communicating a first port to said client terminal as a port for communication, and setting a port used by said server as a second port having a port number other than said publicly known port number; and

conducting data communication with said client terminal using said dedicated protocol via said network, said firewall, and said proxy server.

6. A computer program product, according to claim 5, wherein said program effecting the process of:

conducting the encryption and decoding of data in said data communication.

* * * * *

US 20020078198A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2002/0078198 A1**

Buchbinder et al. (43) **Pub. Date: Jun. 20, 2002**

(54) **PERSONAL SERVER TECHNOLOGY WITH FIREWALL DETECTION AND PENETRATION**

(76) Inventors: **John E. Buchbinder**, Orinda, CA (US); **Alan D. Finke**, Pleasanton, CA (US); **Joshua E. Buchbinder**, Orinda, CA (US)

Correspondence Address:
Geoffrey T. Staniford
DERGOSITS & NOAH LLP
Suite 1450
Four Embarcadero Center
San Francisco, CA 94111 (US)

(21) Appl. No.: **10/077,105**

(22) Filed: **Feb. 15, 2002**

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/513,550, filed on Feb. 25, 2000.

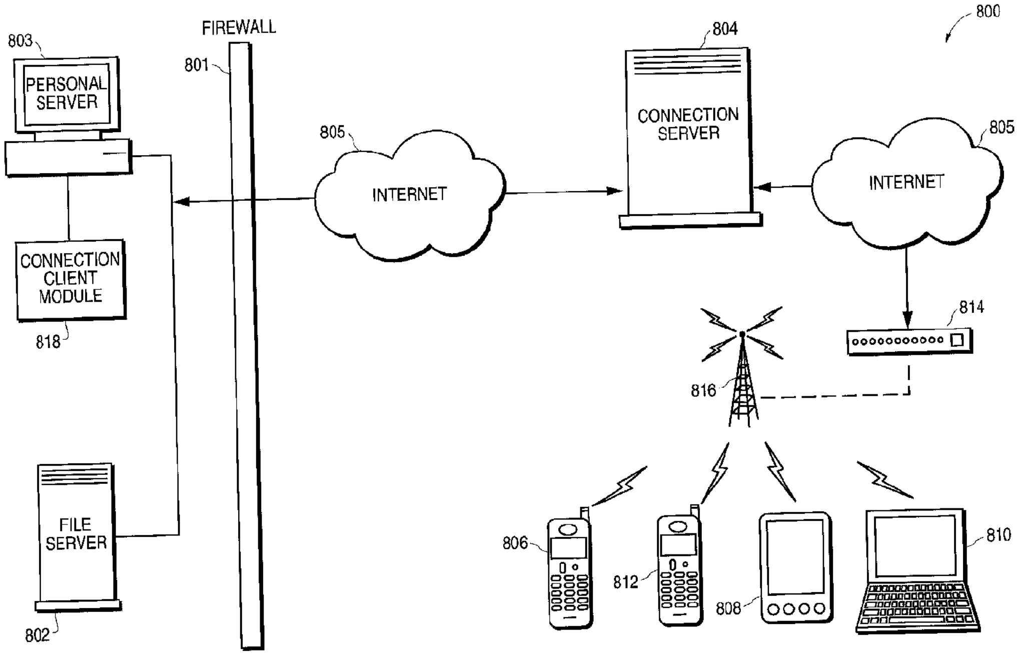
Publication Classification

(51) **Int. Cl.^{7}** **G06F 15/173**; G06F 15/16

(52) **U.S. Cl.** **709/224**; 709/219

(57) **ABSTRACT**

A firewall penetration scheme is described for communication between two networked computers. A first computer within a firewall protected network initiates a connection to a second computer. The second computer is coupled to a network of remote clients that are configured to access the first computer. The first computer transmits a message to the second computer commanding the second computer to connect back to the first computer A series of tests using communication protocols of increasing complexity are executed until a communication protocol enabling communication between the first and second computers is determined. If the address of the first computer changes upon connection, the second computer registers the new address upon each change. If the connection between the first computer and second computer is unintentionally broken, the first computer re-establishes contact with the second computer and maintains the connection by transmitting periodic signals to the second computer.



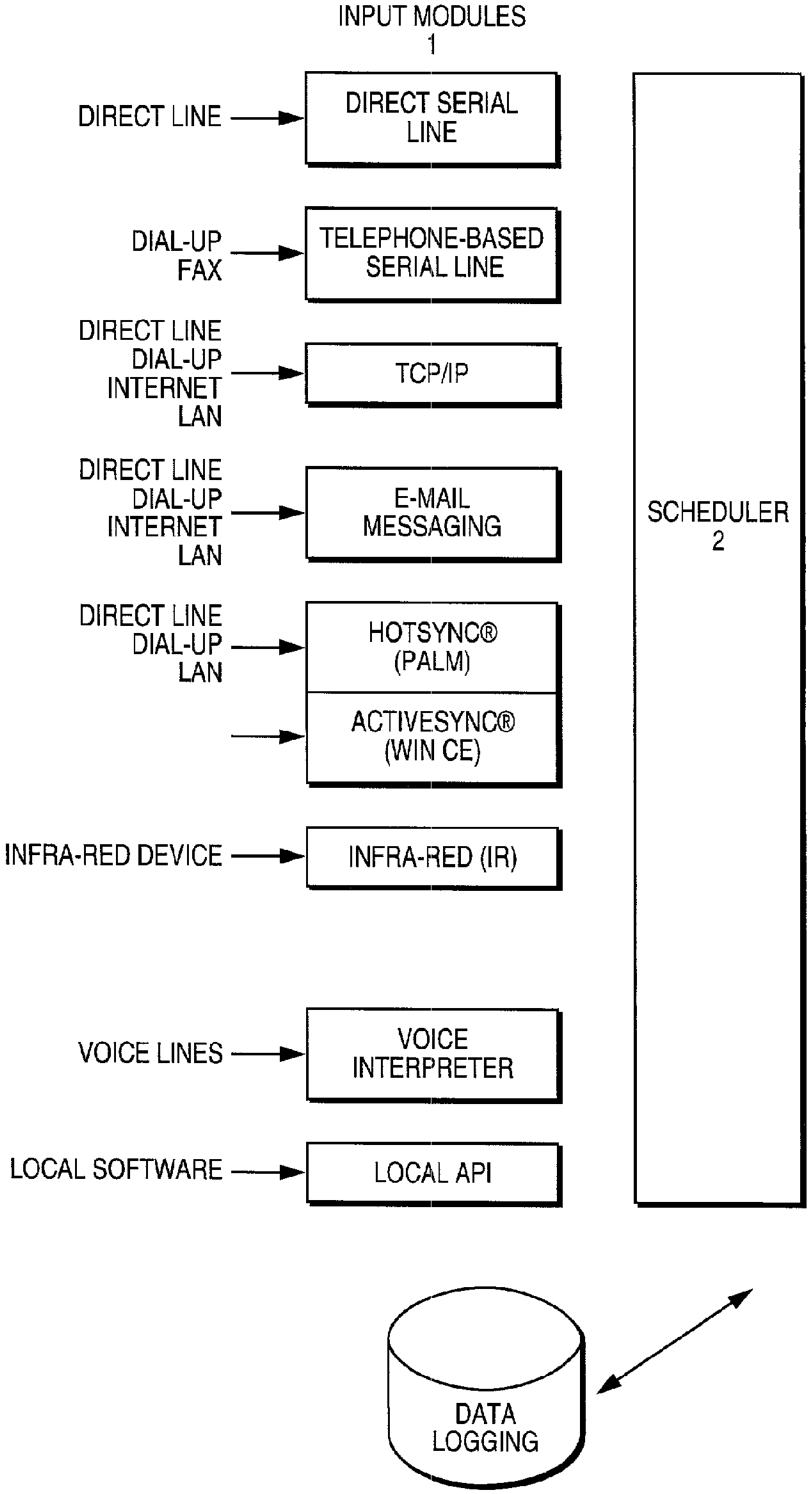

INPUT MODULES
1
DIRECT LINE
DIRECT SERIAL LINE
DIAL-UP
FAX
TELEPHONE-BASED SERIAL LINE
DIRECT LINE
DIAL-UP
INTERNET
LAN
TCP/IP
DIRECT LINE
DIAL-UP
INTERNET
LAN
E-MAIL MESSAGING
DIRECT LINE
DIAL-UP
LAN
HOTSYNC® (PALM)
ACTIVESYNC® (WIN CE)
INFRA-RED DEVICE
INFRA-RED (IR)
VOICE LINES
VOICE INTERPRETER
LOCAL SOFTWARE
LOCAL API
SCHEDULER
2
DATA LOGGING

FIG.1A

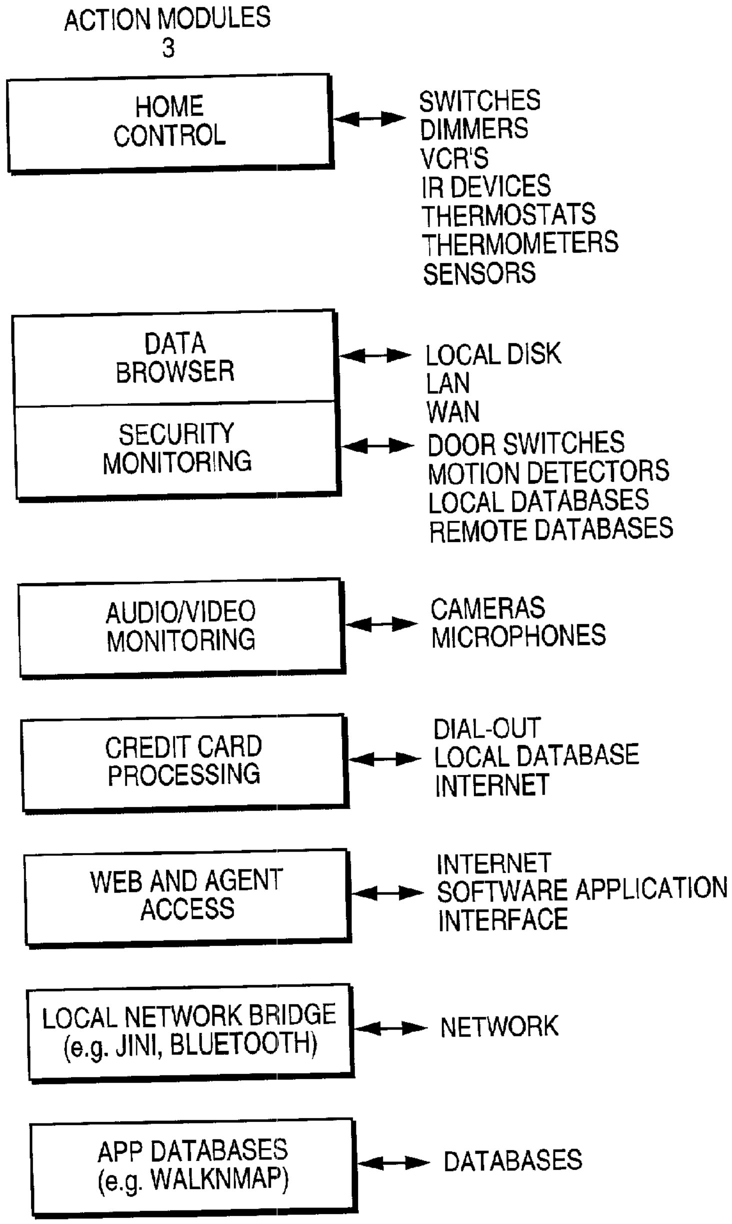

ACTION MODULES
3
HOME CONTROL
SWITCHES
DIMMERS
VCR'S
IR DEVICES
THERMOSTATS
THERMOMETERS
SENSORS
DATA BROWSER
LOCAL DISK
LAN
WAN
SECURITY MONITORING
DOOR SWITCHES
MOTION DETECTORS
LOCAL DATABASES
REMOTE DATABASES
AUDIO/VIDEO MONITORING
CAMERAS
MICROPHONES
CREDIT CARD PROCESSING
DIAL-OUT
LOCAL DATABASE
INTERNET
WEB AND AGENT ACCESS
INTERNET
SOFTWARE APPLICATION
INTERFACE
LOCAL NETWORK BRIDGE (e.g. JINI, BLUETOOTH)
NETWORK
APP DATABASES (e.g. WALKNMAP)
DATABASES

FIG.1B

CONNECTIVITY

THERE ARE MANY MODALITIES FOR CONNECTIVITY, SOME OF WHICH ARE LISTED BELOW.

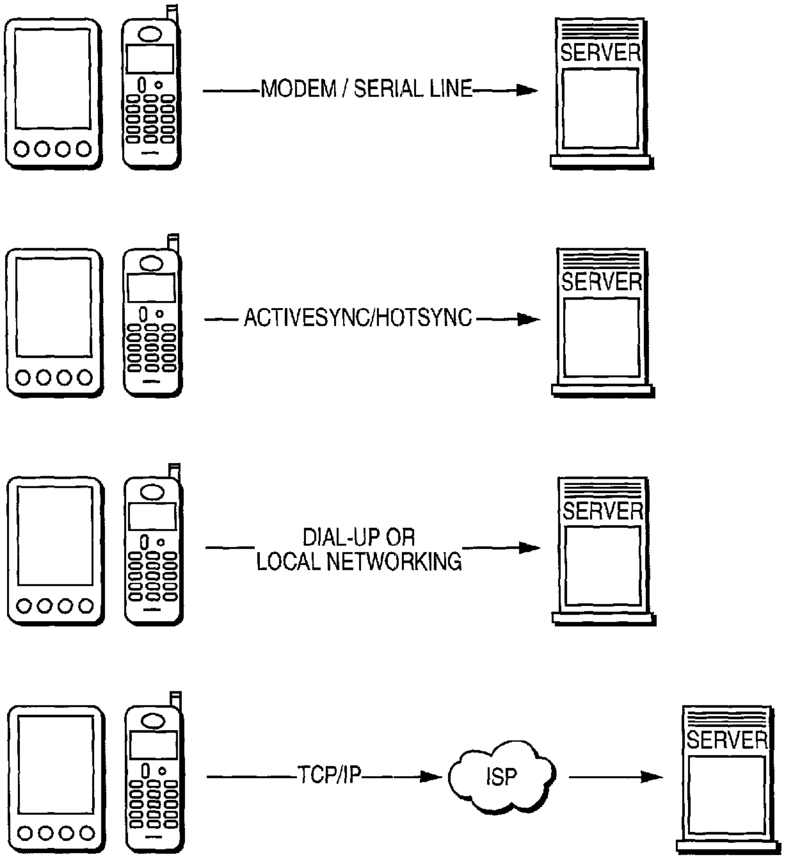


FIG.2A

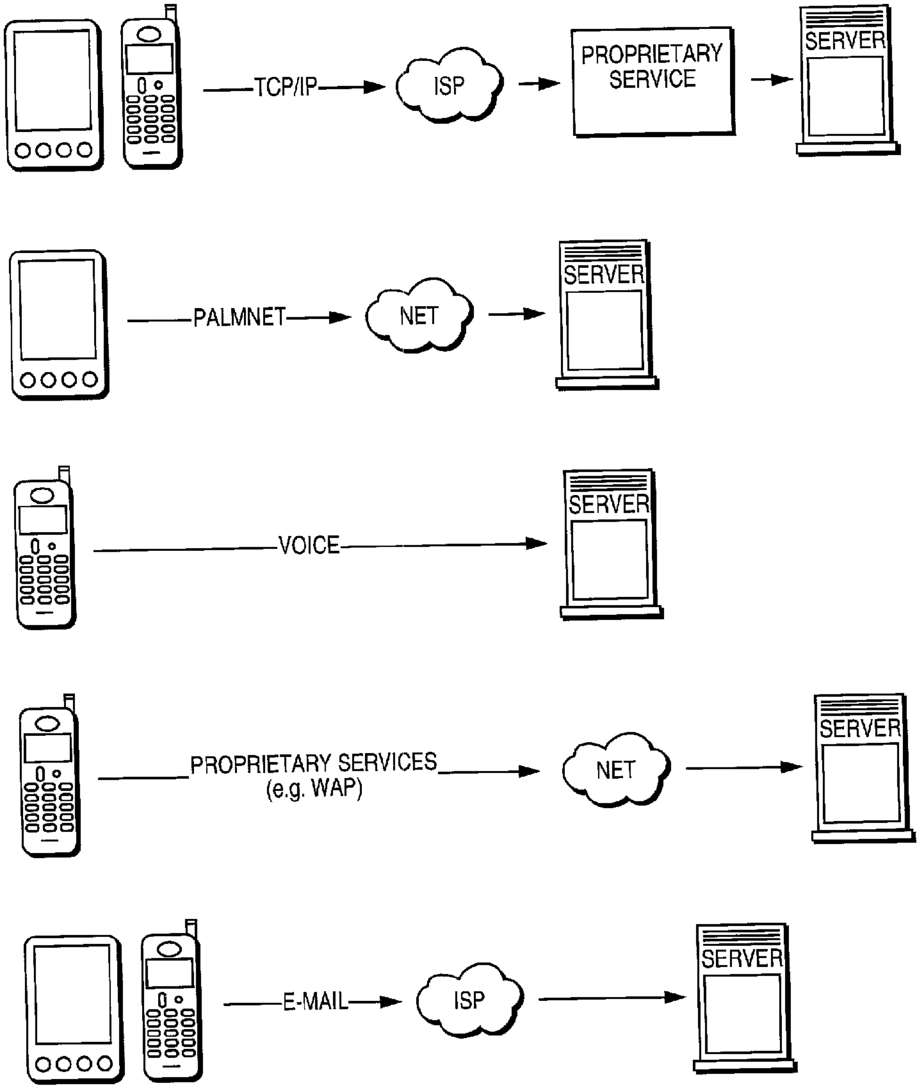
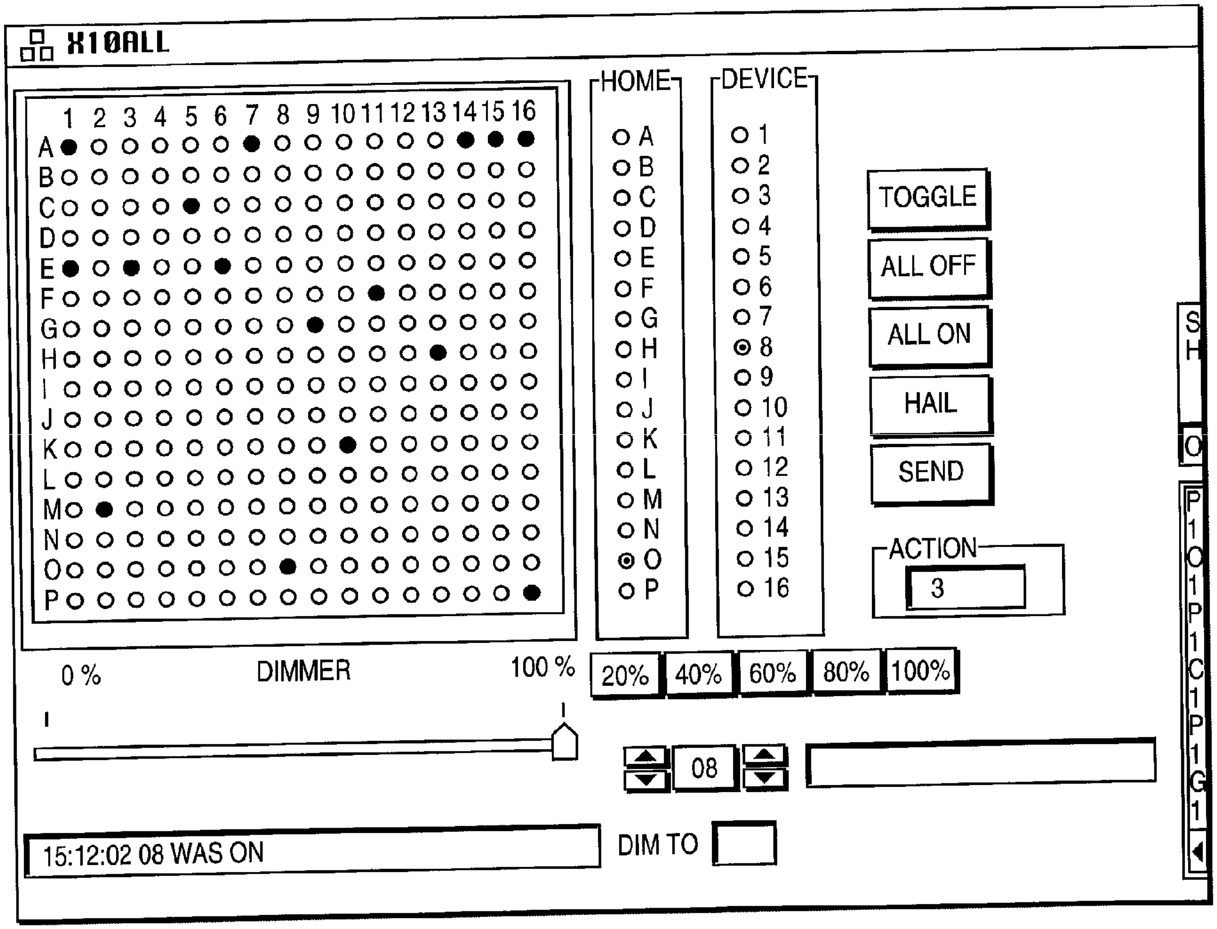

TCP/IP
ISP
PROPRIETARY SERVICE
SERVER
PALMNET
NET
SERVER
VOICE
SERVER
PROPRIETARY SERVICES (e.g. WAP)
NET
SERVER
E-MAIL
ISP
SERVER

FIG.2B



X10ALL

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	●	○	○	○	○	○	●	○	○	○	○	○	○	●	●	●
B	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
C	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○
D	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
E	●	○	●	○	○	●	○	○	○	○	○	○	○	○	○	○
F	○	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○
G	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○
H	○	○	○	○	○	○	○	○	○	○	○	○	●	○	○	○
I	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
J	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
K	○	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○
L	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
M	○	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
O	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○
P	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●

HOME
○ A
○ B
○ C
○ D
○ E
○ F
○ G
○ H
○ I
○ J
○ K
○ L
○ M
○ N
⊙ O
○ P

DEVICE
○ 1
○ 2
○ 3
○ 4
○ 5
○ 6
○ 7
⊙ 8
○ 9
○ 10
○ 11
○ 12
○ 13
○ 14
○ 15
○ 16

ACTION: 3

DIMMER 0 % – 100 %

15:12:02 08 WAS ON

DIM TO

08

FIG.3

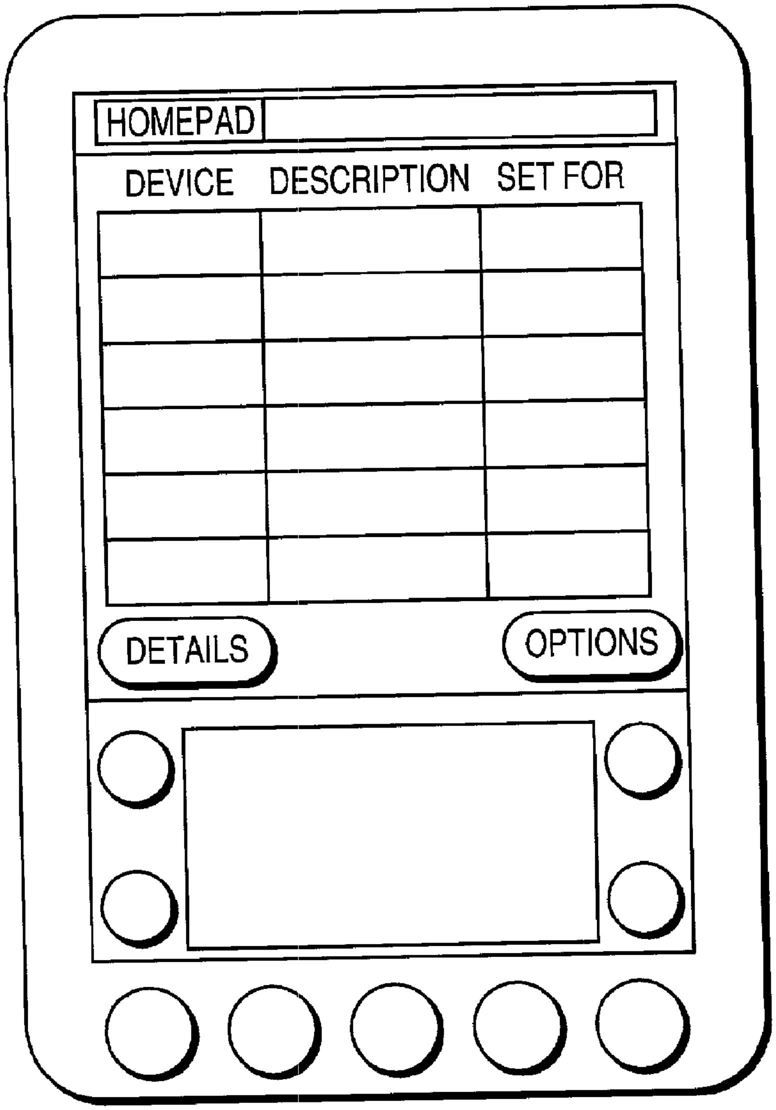


FIG.4A

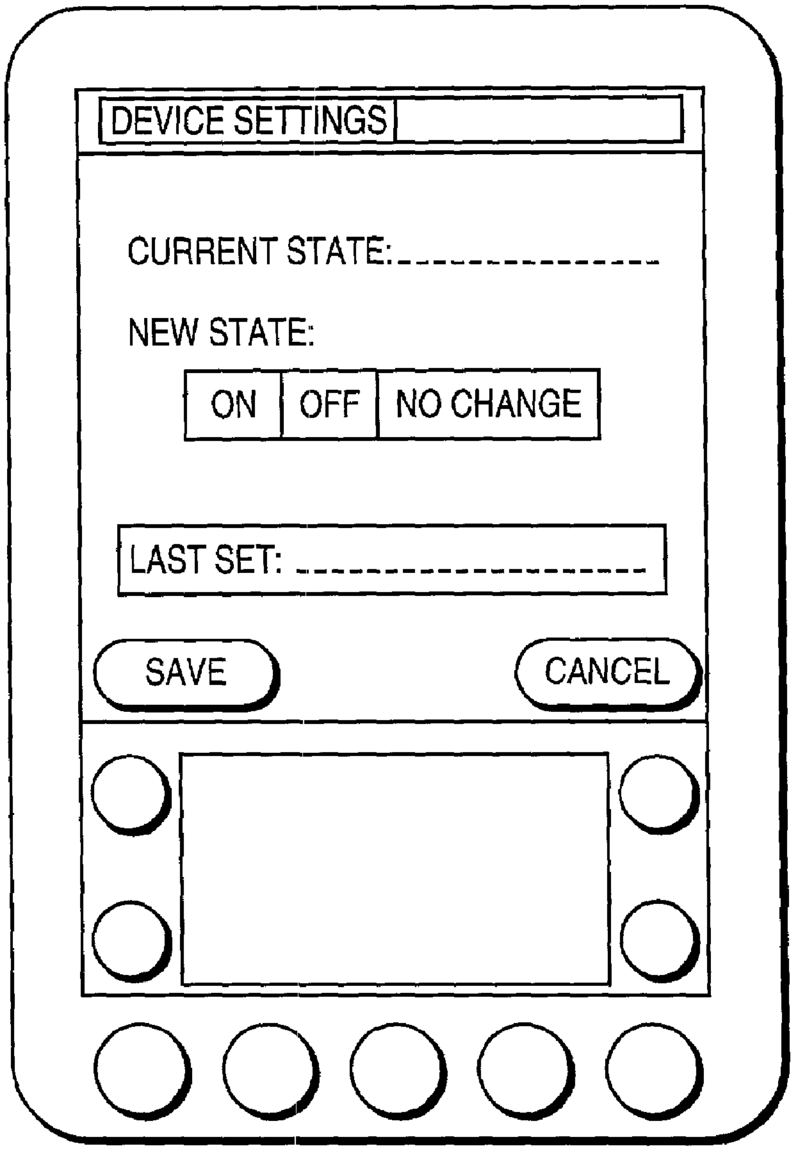

DEVICE SETTINGS
CURRENT STATE:
NEW STATE:
ON
OFF
NO CHANGE
LAST SET:
SAVE
CANCEL

FIG.4B

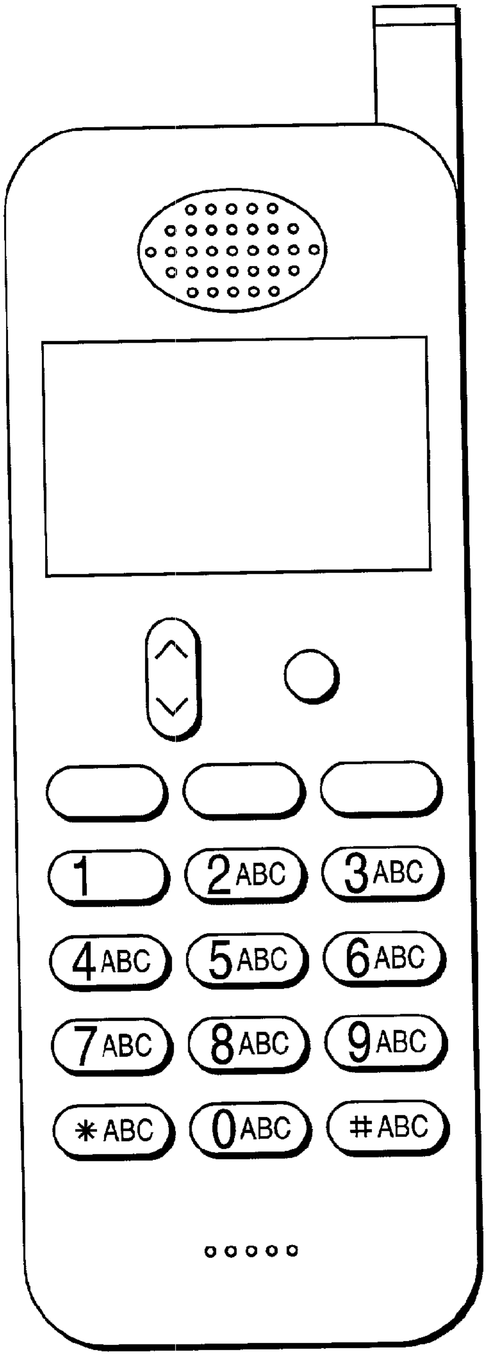


FIG.5

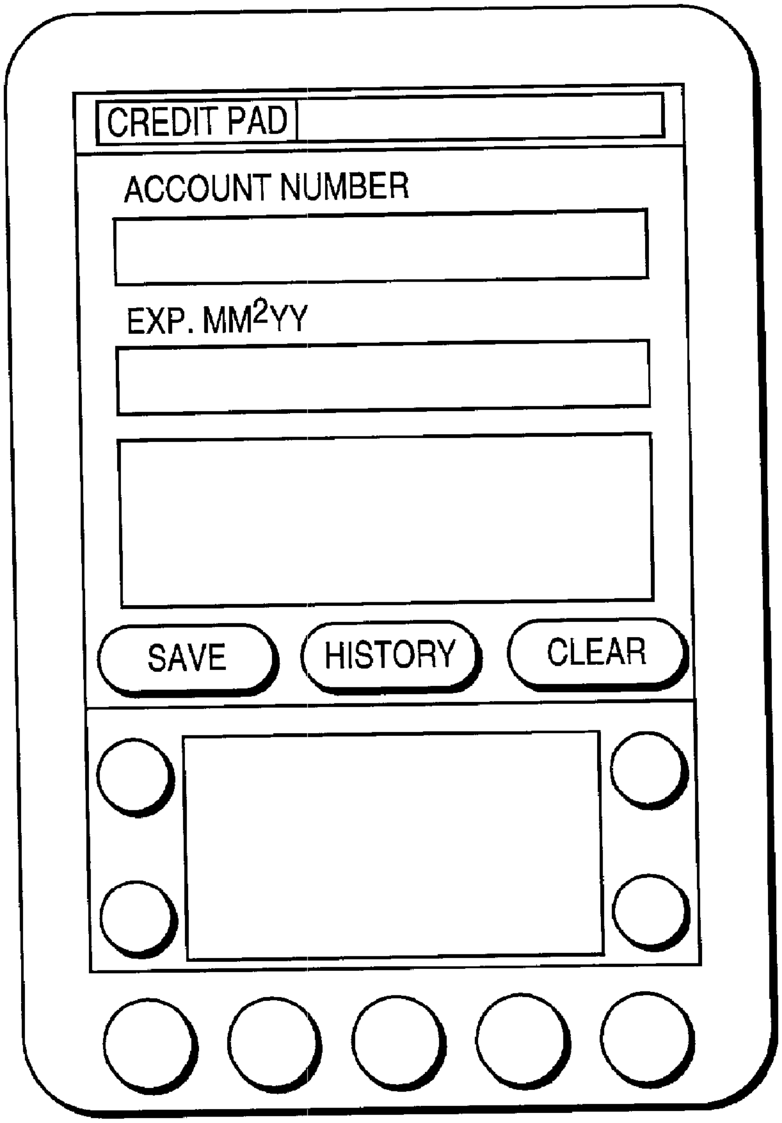
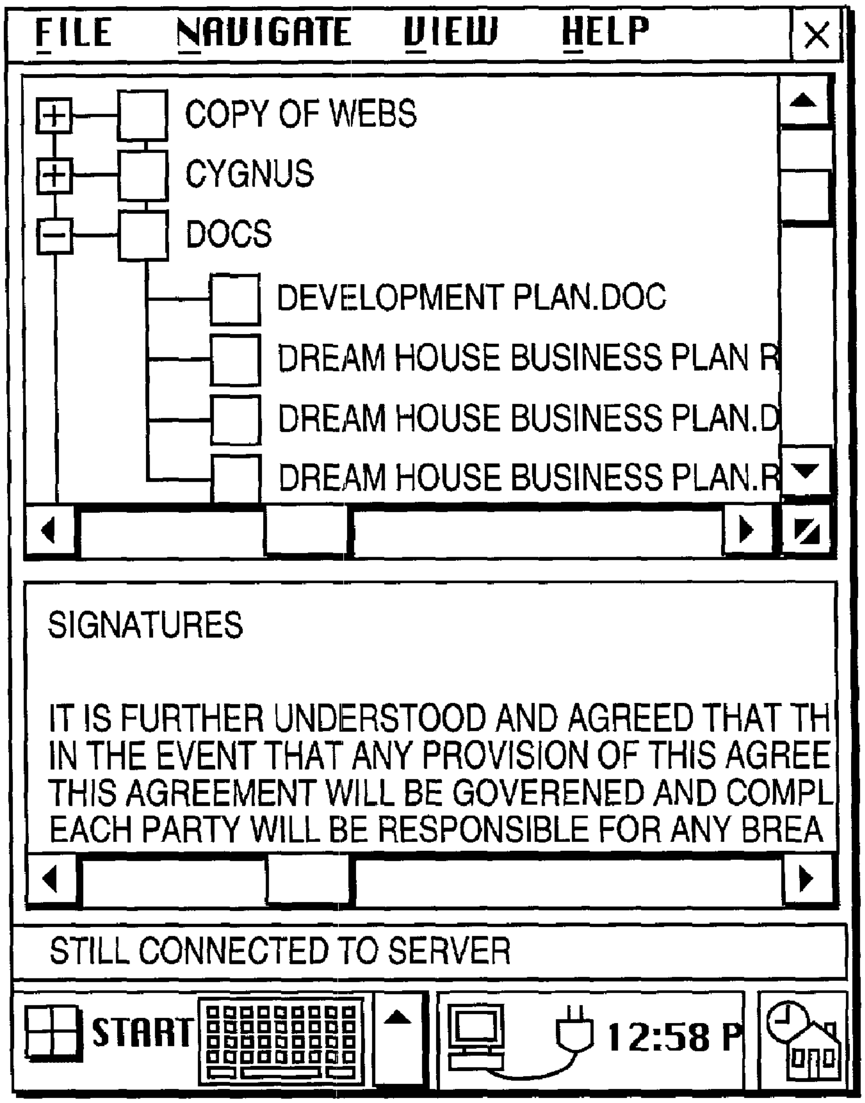


FIG.6



COPY OF WEBS
CYGNUS
DOCS
DEVELOPMENT PLAN.DOC
DREAM HOUSE BUSINESS PLAN R
DREAM HOUSE BUSINESS PLAN.D
DREAM HOUSE BUSINESS PLAN.R

SIGNATURES

IT IS FURTHER UNDERSTOOD AND AGREED THAT TH
IN THE EVENT THAT ANY PROVISION OF THIS AGREE
THIS AGREEMENT WILL BE GOVERENED AND COMPL
EACH PARTY WILL BE RESPONSIBLE FOR ANY BREA

STILL CONNECTED TO SERVER

FIG.7

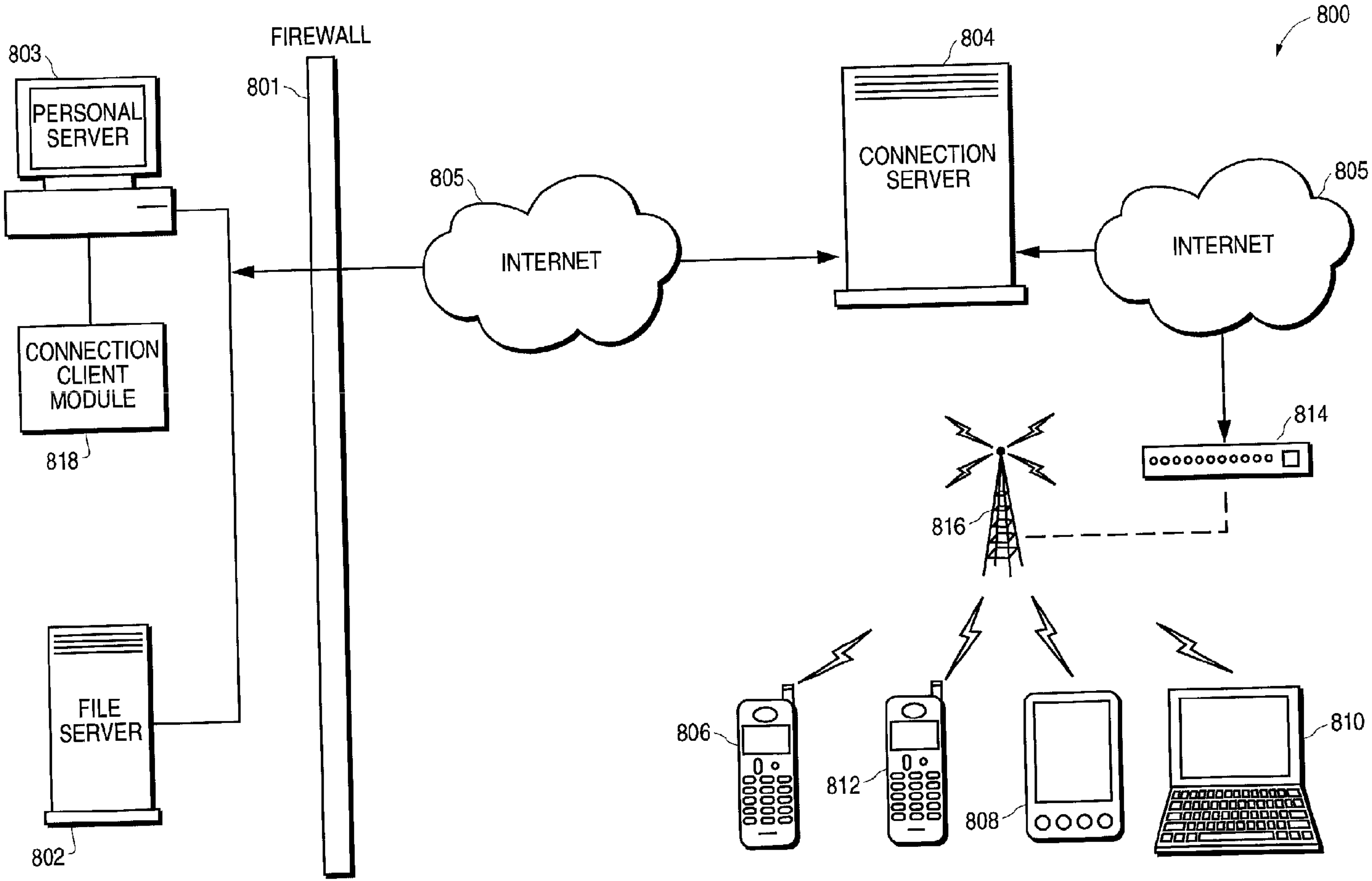

800
803
PERSONAL SERVER
FIREWALL
801
805
INTERNET
804
CONNECTION SERVER
805
INTERNET
CONNECTION CLIENT MODULE
818
814
816
FILE SERVER
802
806
812
808
810

FIG.8A

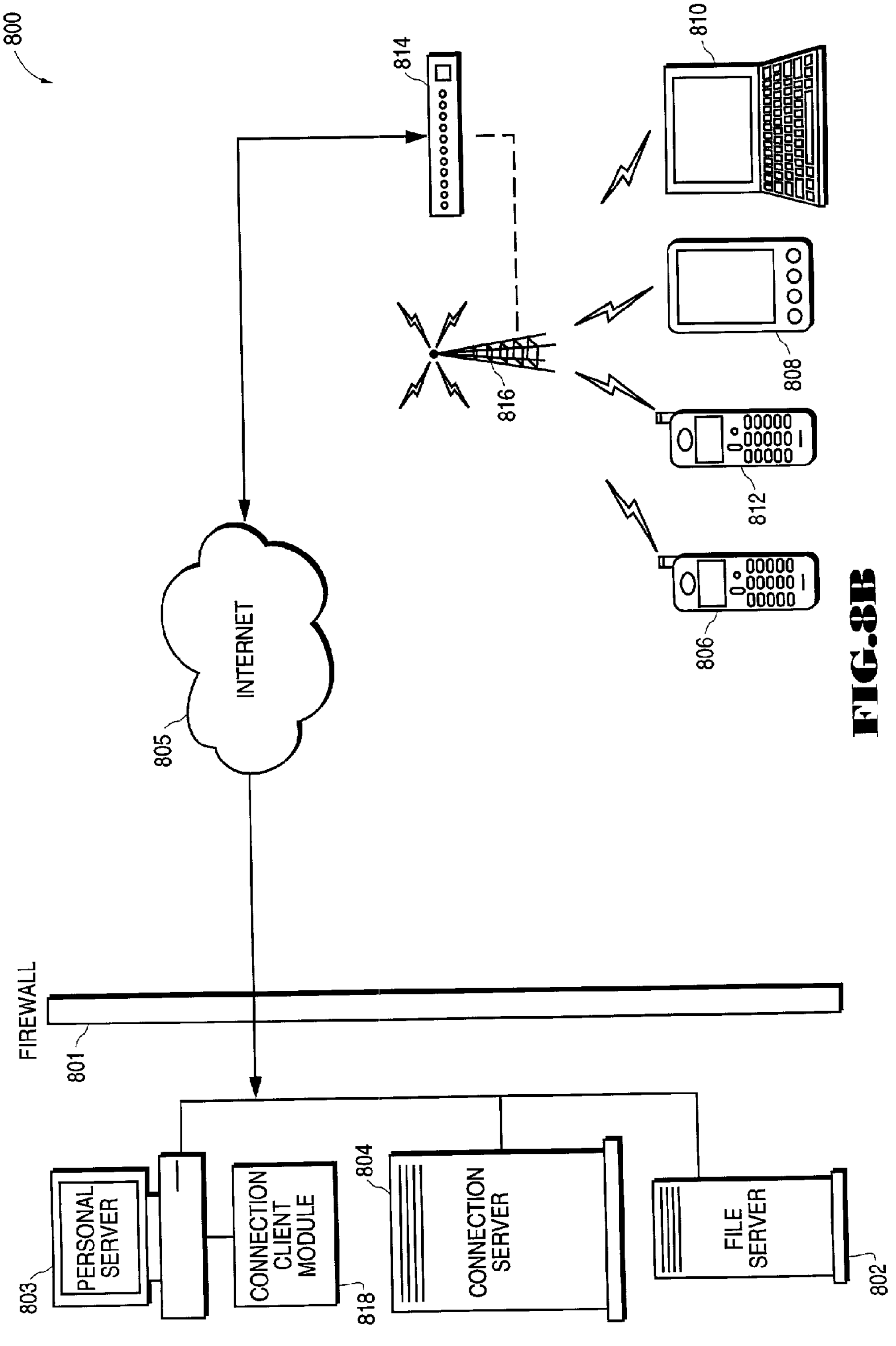

800
814
810
816
808
812
806
INTERNET
805
FIREWALL
801
PERSONAL SERVER
803
CONNECTION CLIENT MODULE
818
804
CONNECTION SERVER
FILE SERVER
802

FIG.8B

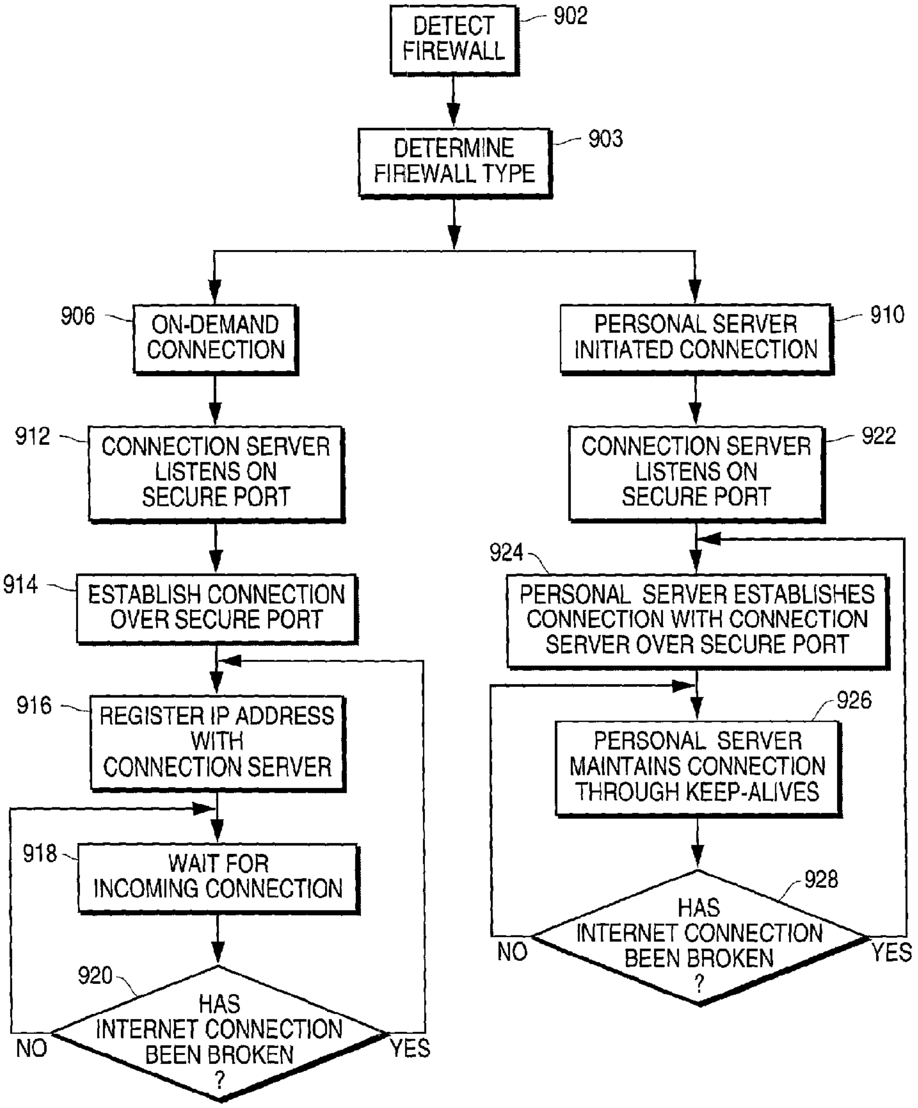

DETECT FIREWALL
902
DETERMINE FIREWALL TYPE
903
906
ON-DEMAND CONNECTION
PERSONAL SERVER INITIATED CONNECTION
910
912
CONNECTION SERVER LISTENS ON SECURE PORT
CONNECTION SERVER LISTENS ON SECURE PORT
922
914
ESTABLISH CONNECTION OVER SECURE PORT
924
PERSONAL SERVER ESTABLISHES CONNECTION WITH CONNECTION SERVER OVER SECURE PORT
916
REGISTER IP ADDRESS WITH CONNECTION SERVER
926
PERSONAL SERVER MAINTAINS CONNECTION THROUGH KEEP-ALIVES
918
WAIT FOR INCOMING CONNECTION
928
NO
HAS INTERNET CONNECTION BEEN BROKEN ?
YES
920
NO
HAS INTERNET CONNECTION BEEN BROKEN ?
YES

FIG.9

PERSONAL SERVER TECHNOLOGY WITH FIREWALL DETECTION AND PENETRATION

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application is a continuation-in-part application of U.S. application Ser. No. 09/513,550, entitled "PERSONAL SERVER TECHNOLOGY", filed on Feb. 25, 2000.

FIELD OF THE INVENTION

[0002] The present invention relates generally to computer device networks, and more specifically to a wireless local area network that integrates home appliances, computing devices, and other objects into a coordinated wireless control and monitoring network, and that provides penetration of protection mechanisms within the local area network.

BACKGROUND OF THE INVENTION

[0003] Systems that monitor and control electronic appliances and other objects in the home and office are known. Such systems, however, are limited almost exclusively to "remote control" systems involving the use of a hand-held device to send instructions directly to and receive information directly from one, or at most a few, objects. One example of such a remote control device is the standard VCR (video cassette recorder) remote, which operates on infrared (IR) light wavelengths. A VCR remote is typically used to program recording parameters into a VCR and to operate the VCR in real-time. Similar remote control devices exist for TVs, CD players and other appliances. Lights and other household fixtures can also be controlled by remote, usually by installation of a component that allows for simple commands such as on/off and dimming in response to hardwired timers, audible input, or other control means.

[0004] However, the state of remote control of home appliances and electronic equipment in the current art is nascent. Some objects such as VCRs and CD (compact disk) players usually have remote control devices, but many do not. Even among the objects that do have remote control, such objects are not controlled through integrated networks. In fact, the notion of a connectivity system or solution hardly applies to the state of the current art. Of the relatively few objects in a present-day home or office that can be controlled by remote, each one generally requires a separate remote control device. Sometimes, a handful of objects (e.g., CD player, amplifier and tuner) can be controlled with a single remote from a single manufacturer of the devices, or they can be standardized to a single "universal" remote that can control a large number of TVs and VCRs.

[0005] Some present systems include home control systems that allow a user to control lights, sound systems, and other fixtures throughout the household. While appearing to be along the lines of a true "control network," these systems still exhibit only rudimentary control over and feedback from objects that are connected to the network. In addition, these systems are difficult to implement, and do not offer the power and flexibility of a programmable, software-based network. They also cannot be controlled and monitored from outside the home via network and Internet connections.

[0006] The true networks that do exist in the current art are essentially limited to information exchange. For instance, U.S. Pat. No. 5,809,415, issued to Rossmann, which is herein incorporated by reference in its entirety, describes a two-way, portable data-communication device that allows user access to a wide-area network, such as the Internet. Such inventions are limited in the opposite way that home-control and remote-control systems are limited. The former cannot manipulate and monitor the physical devices, at least not to any appreciable degree, while the latter lack the information, control and integration aspects of a true network.

[0007] For these reasons, among others, there is a need in the art for a true network that can bring a large number of objects under the control of a single, integrated connectivity solution. This solution would ideally be flexible enough to be easily programmed for different network configurations and settings, and powerful enough to allow the user to have precise control and perception of the objects in the network through the metaphor of an intuitive user interface.

[0008] A further disadvantage associated with present systems for networking home control systems is the inability to effectively accommodate network security structures, such as firewalls and other network filters. In a computer network, a firewall can be implemented as a single router that filters out unwanted communication packets, or it may comprise a combination of routers and servers each performing some type of firewall processing. Firewalls are widely used to give users secure access to the Internet and to keep internal network segments secure. However, in certain situations, these firewalls also prevent desired access from one network to another. Present systems of networking devices in a home control environment generally cannot penetrate firewall protected networks. This limits the use of present home control environments from effectively allowing access and control to other networks, such as the Internet.

[0009] Although generic firewall bridge systems do exist for allowing network access through firewall protected computers, these systems typically require the implementation of a Virtual Private Network (VPN), or private dedicated lines necessary for security. The use of VPN technology is generally disadvantageous because implementation is often difficult and expensive, and requires high maintenance Present VPN systems also suffer from the drawback of generally not working with Personal Digital Assistant (PDA) devices, thus limiting their effectiveness in wireless network systems.

SUMMARY OF THE INVENTION

[0010] A connectivity system for use in the home, office and other locations that incorporates a method of penetrating fireball protection schemes is described. The system comprises a server-like apparatus that integrates home appliances, entertainment systems, computing devices, and other objects into a coordinated wireless control and monitoring network. A remote device is used to control and monitor these objects via the functioning of the server-like apparatus. The server-like apparatus is also connected to other networks, such as the Internet. The remote device presents the user with a powerful, easy-to-use interface environment that intuitively maps to the objects on the network and the actions and activities being performed. The present invention thus implements an automated, intelligent, seamlessly connected "home or office of the future."

[0011] The present invention offers an integrated connectivity solution for remote control of various network integrated household and office objects ("Controlled Devices"). It comprises a software-based network that can perform information-heavy tasks and that incorporates sophisticated object monitoring and control, as well as computational activities, into the network. The present invention consists of a server-like apparatus ("Personal Server") that controls a network, and performs computational tasks, in the home, office, or other location. The Personal Server is accessed through a Remote Device, generally a hand-held, personal digital assistant ("PDA"), a data-enabled telephone or cellular phone ("SmartPhone"), or some form of internet access device. PALM O/S™ devices such as the PALM PILOT™, PALM III™ and PALM IV™, and WINDOWS CE™ devices such as the PHILIPS NINO™, CASIO CASSIOPEIA™ and HP JORDANA™ are common PDAs that are readily adaptable for use with the present invention. The Qualcomm PdQ phone, a cellular phone with digital computing and display capabilities, is an example of a SmartPhone that will work well with the present invention.

[0012] Embodiments of the present invention allow users to control and monitor various Controlled Devices. These functions can be accomplished from within the location where the Personal Server is located, or from the outside world thorough a dial-up connection, network, or the Internet, or other means. Remote information tasks, such as file exchange, computational activity and financial transactions can also be carried out by the Personal Server, using a Remote Client operating on a Remote Device as the interface. Third parties, such as alarm companies and police departments, can be given full or partial access to the monitoring and control functions of the Personal Server.

[0013] Embodiments of the present invention also allow penetration of firewalls and other protection devices between the Personal Server and the Controlled Devices. A connection module within the Personal Server establishes communication with a Connection Server, which is directly or indirectly coupled to one or more Controlled Devices. The connection module determines the type of firewall that exists between the user computer and the Personal Server. Protection protocols of increasing complexity are tested until the type of firewall is determined. This protocol is then used for subsequent communication. If the address of the Personal Server is dynamic, the Personal Server registers its new address with the Connection Server upon each connection. The Connection Server then tracks the address of the user computer. If the connection between the Connection Server and Personal Server is unintentionally broken, the Personal Server re-establishes communication, and transmits periodic "keep alive" signals to the Connection Server to maintain the connection.

[0014] Other objects, features, and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description that follows below.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings, in which like references indicate similar elements, and in which:

[0016] **FIG. 1** illustrates a personal server, including Action Modules, Scheduler/Router, and Input/Output Modules, according to one embodiment of the present invention;

[0017] **FIG. 2** illustrates some examples of the physical connection and data transfer protocols that can be used between the Remote Device and the Personal Server;

[0018] **FIG. 3** shows a control panel that is used to configure the network of objects on the Personal Server, according to one embodiment of the present invention;

[0019] **FIGS. 4A and 4B** show an example of a screen on the Remote Client interface running on the Remote Device that can be used in conjunction with embodiments of the present invention;

[0020] **FIG. 5** shows an embodiment of Home Pad on a more graphically limited Remote Device, namely, a cell phone;

[0021] **FIG. 6** shows a second example of a screen on the Remote Client interface running on the Remote Device used with the present invention, in this case, Credit Pad;

[0022] **FIG. 7** shows a third example of a screen on the Remote Client interface running on the Remote Device used with the present invention, in this case, File Retriever;

[0023] **FIG. 8A** illustrates a Personal Server network that includes a firewall detection and penetration scheme, according to one embodiment of the present invention;

[0024] **FIG. 8B** illustrates a Personal Server network that includes a firewall detection and penetration scheme, according to an alternative embodiment of the present invention; and

[0025] **FIG. 9** is a flowchart that illustrates the method of identifying the presence of a firewall and establishing a communication conduit between a Personal Server and a Connection Server coupled to a Remote Device, according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0026] A wireless personal server for interfacing a variety of home appliance and computing devices in a firewall protected network environment is described. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be evident, however, to one of ordinary skill in the art, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form to facilitate explanation. The description of preferred embodiments is not intended to limit the scope of the claims appended hereto.

[0027] Aspects of the present invention may be implemented on one or more computers executing software instructions. According to one embodiment of the present invention, server and client computer systems transmit and receive data over a computer network, standard telephone line, or wireless data link. The steps of accessing, downloading, and manipulating the data, as well as other aspects of the present invention are implemented by central processing units (CPU) in the server and client computers executing sequences of instructions stored in a memory. The memory may be a random access memory (RAM), read-only memory (ROM), a persistent store, such as a mass storage device, or any combination of these devices. Execution of the sequences of instructions causes the CPU to perform steps according to embodiments of the present invention.

[0028] In a preferred embodiment, the core of the present invention is a server-like apparatus ("Personal Server"). The Personal Server comprises software run on a general-purpose computer. The computer can be a server, workstation, dedicated hardware device, or any other type of computer. In the description that follows, it is assumed that the computer comprising the Personal Server is a desktop PC. In other embodiments, the Personal Server comprises hardware specifically designed for the invention, or a combination of hardware and computer software. The software can be a component bought off the shelf, a component specially designed for a particular home or office, a plug-in to a software developer's kit, or part of a larger proprietary system, among other embodiments. The software of the Personal Server is typically written in C, C++or Java™. The Personal Server is designed to have a robust and flexible interface that makes it easy for developers to develop Input/Output and Action Modules that operate with the present invention.

[0029] 1. Software Architecture

[0030] a. Personal Server

[0031] The following is a preferred embodiment of the software architecture of the present invention. **FIG. 1** illustrates a further preferred embodiment, in detail.

[0032] The Personal Server has a software architecture that consists of the following components: Input/Output Modules **1**, a core Scheduler/Router **2** with data logging capabilities and Action Modules **3**. The Input/Output Modules 1 and Action Modules 3 are self-contained code libraries designed to be detected by the Scheduler/Router **2** and connected at run-time. This architecture allows developers and consultants to develop additional modules, either for a class of users or Controlled Devices, or on a case-by-case basis for specific individual users, to fit those users needs. In particular, as new forms of communication, types of Controlled Devices, and activity are developed through technological development and commercial innovation, new types of modules will be developed. Such modules can be added to the Personal Server by direct installation or by downloading on an ad-hoc basis from remote sources. They can also be dynamically added to individual installations of the Personal Server, with or without user intervention, to minimize service interruption.

[0033] Input/Output Modules **1** serve to connect a user's Remote Device to the Personal Server, but they can be designed for other modes of communication as well. Various types of physical connections and data-transfer protocols can be used, as illustrated in **FIG. 2**. At synchronization, the Remote Device sends the information entered by the user to an Input/Output Module or Input/Output Modules. This information is translated into a "Message" by the Input/Output Module. Messages generally contain information on the user, the Remote Device, the target Action Module and data specifics. The Message may be encoded or encrypted for the purpose of data security. In one encryption scheme, Messages are encrypted by the Remote Device prior to transmission, and then decoded by the Input/Output Module. The Input/Output Module then passes the Message to the Scheduler/Router, which logs it into a database, processes it as necessary, and passes the Message again to the appropriate Action Module. The Action Module then performs the requested actions. After the action has been completed, the Action Module creates a second Message containing user-requested information, results of calculations or computations, information on whether the action has been successfully completed, date and time stamps, and whether additional instructions are needed. The Action Module passes the Message to the Scheduler/Router, which logs it, processes it as necessary, and passes it, if necessary, to the Input/Output Module. The Input/Output Module then communicates the Message contents, possibly in encrypted format, to the Remote Device. Additional messages not specifically mentioned may be created and sent as particularly in other embodiments. Alternate embodiments employ separate Input Modules and Output Modules rather than combined Input/Output Modules. In such alternate embodiments, Input Modules are responsible for receiving Messages from the Remote Device, whereas Output Modules are responsible for sending Messages to the Remote Device.

[0034] At start-up, the Scheduler/Router loads the existing Input/Output Modules and Action Modules and monitors them for activity. As noted, the Scheduler/Router processes and relays Messages between the Input/Output and Action Modules. It maintains information on user identification, user password and security information, as well as logs of the Messages. In a preferred embodiment, a Utility Module is written as an adjunct to the Scheduler/Router, which allows the user to enter settings. The Utility Module will generally have a control-panel type interface to aid in configuring new user preferences and new modules.

[0035] The Action Modules or the Scheduler/Router may initiate messages to the user. If the user has requested an action to be performed that may take a long time, the user may disconnect and request that the results be sent back at a later time. Alternately, a Controlled Device may initiate a communication, triggering an Action Module to send a Message to the Scheduler/Router. In this way, the user may configure the system so that the Personal Server initiates communication when triggered by an event such as a home alarm being set off. Results may be sent back when the user connects again, by a connection established by the Personal Server, or by another communication means such as pager, telephone, fax, or e-mail.

[0036] b. Input/Output Modules

[0037] As described in the section above, Input/Output Modules 1 serve as connection points between the Personal Server and the Remote Device. The various Input/Output Modules in place with a particular embodiment of the Personal Server are designed to handle various connectivity and data-transfer protocols (some examples of which are listed in **FIG. 2**). In a preferred embodiment, proprietary PDAs protocols such as HOTSYNC™ (for PALM OS™ devices) and ACTIVESYNC™ (for WINDOWS CE™ devices) are among these protocols. In the case of incoming Messages an Input/Output Module communicates with a Remote Device by synchronizing with the Remote Device, receiving and interpreting a Message from the Remote

Device, optionally decrypting the Message if it is in encrypted form, and then passing the Messages on to the Scheduler/Router which in turn optionally passes that Message in original or modified form on to an Action Module and possibly a Controlled Device. In the case of outgoing Messages an Input/Output Module communicates with a Remote Device by synchronizing with the Remote Device, receiving and interpreting a Message from the Scheduler/Router (which Message may have originated from a Controlled Device or Action Module), optionally encrypting the Message, and then passing the Messages on to the Remote Device, which in turn decrypts the Message as necessary.

[0038] In alternate embodiments connection to the Input/Output Modules may be mediated by an Internet service designed specifically to communicate with the Personal Server, or else to a general-purpose Internet service (the "Service"). The user operating the Remote Device may log in or otherwise connect to the Service. In either event, the user accesses a network server (the "Internet Server") which runs the Service via a website or other user interface. Once the user has logged in using a Remote Device, the Service will then complete the final link to the Personal Server. The Service may dial-in, or use any of the means of connectivity supported by the Input/Output Modules, and then communicate with the Personal Server using standard protocols. The Messages from the Personal Server are then communicated back to the user. Thus a user can use a Remote Device such as a Web-enabled cellular phone to connect to a Personal Server at home or at the workplace.

[0039] In alternate embodiments there may be no encryption provided, or the encryption/decryption function may occur at different locations on the system such as at the Scheduler/Router, Action Module, or Controlled Device rather than or in addition to the encryption provided by the Input/Output Module. In other alternate embodiments encryption/decryption functions may occur at the level of the Remote Client or the Service rather than or in addition to the encryption provided by the Remote Device.

[0040] c. Action Modules

[0041] The Action Modules are the software objects that actually carry out instructions specified by the user, and that obtain status and other information from and send instructions to the Controlled Devices. Because of the wide variety of specific actions they carry out, Action Modules will often include their own databases to assist in their functions. Some Action Modules will have their own connectivity to the Web and to other communication lines. An Action Module may be connected to a third party or parties, to the Internet, to other computer systems, or to other networks (even other Personal Server networks).

[0042] d. Messages

[0043] In a preferred embodiment Input/Output Module some Messages from the Input/Output Module to the Scheduler/Router comprise user information, intended Action Module or modules, message length, time stamp and data specifics The data specifics contain specific commands to the Action Module or Action Modules such as requests for state information as well as any data needed by the Action Module to perform its tasks.

[0044] Messages from the Scheduler/Router to the Input/Output Module comprise user information, Action Module identification, message length, time stamp, and data specifics. The data specifics contain responses requested by the user, the results of actions performed, state information, response formatting information, and possible requests for additional information from the input device.

[0045] In alternate embodiments, Messages may originate or terminate, or be interpreted, parsed, decoded, encoded, modified, scheduled, or otherwise processed by the Remote Client, the Remote Device, the Service, the Input/Output Module, the Scheduler/Router, the Action Module, or the Controlled Device. New Input/Output Modules and message protocols can be developed by one of ordinary skill in the art as new technologies, in particular O/S device types, are developed.

[0046] e. Remote Client/Remote Device

[0047] The Remote Client is the user's interface and architecture for the Personal Server. It resides on the Remote Device as a data-gathering/presentation medium. The Remote Device, in a preferred embodiment, is a handheld PDA such as a PALM O/S™ WINDOWS CE™ device, or SmartPhone. In alternate embodiments the Remote Device may be a desktop personal computer or any form of Internet access device. Since many Remote Devices, especially handheld devices, are limited in terms of processing power, memory and display capabilities, the Remote Client is generally designed with these limitations in mind. Therefore, in a preferred embodiment, the software architecture of the present invention relies most heavily on the Personal Server itself, rather than on the Remote Client. In some embodiments, a laptop or even desktop computer will act as the Remote Device, often connected through a network, such as the Internet, but even in these cases, the degree of input available from the computer may be limited. In addition, a web page served by a mediating Service on the Internet may serve as the interface for communication to the user. This allows limited input through an Internet access device such as a SmartPhone or Internet kiosk.

[0048] The Remote Client presents an environment that precisely maps to the network of objects to be controlled through the Personal Server, thus allowing seamless control and perception over the network. The Remote Client has the appropriate interfaces, which communicate with the Input/Output Modules of the Personal Server. The Remote Client is generally designed with the most minimal interface environment that nonetheless remains clear and intuitive to the user. FIGS. **4-6** illustrate sample Remote Client environments, including Home Pad, Credit Pad and File Retriever (see "Brief Description of Drawings"). While somewhat less complex than an environment on the Personal Server itself, such as the X10 control interface of **FIG. 3**, Remote Client environments nonetheless remain robust and easy to use.

[0049] The Remote Client also generally uses the minimum amount of encryption and authentication necessary to preserve security. Remote Devices, particularly third-party Remote Devices, will generally be programmed to operate as the Remote Client. Some Remote Devices will be adapted with additional hardware to operate as the Remote Client, and some will be manufactured specifically for use with the present invention.

[0050] Remote Devices may use a variety of physical connection and data transfer protocols to communicate with the Personal Server, some examples of which are illustrated in **FIG. 2**. Typically more than one protocol will be available, depending on where the user and the Remote Device happen to be at the time of linking. The following is another way of categorizing the types of connections:

[0051] 1. Through the same wireless network that is used to control objects in the home or office (used when the user is in or near that home or office)

[0052] 2. Through a different wireless network

[0053] 3. Through a direct wire-based or wireless connection, such as a serial computer interface (used when the Remote Device is "plugged-into" the Personal Server for data transfer or programming

[0054] 4. Through a dial-in modem connection

[0055] 5. Through a dial-up service, Internet service, or other mediating Service on the Internet or other Wide-Area networks

[0056] Traditional phone lines, leased lines and satellite connections are among the communication pipes that can be used to support these physical connections. In some cases, it will be desirable for the user to authorize third-party access to some or all of the control and monitoring systems of the Personal Server. For instance, a user may allow an alarm company to monitor the alarm system. The user may also wish to give some access to a family member or friend if the user is on vacation or otherwise indisposed.

[0057] 2. Method

[0058] a. Direct Connection.

[0059] The following flowchart illustrates, as a preferred embodiment, the method of using a device constructed in accordance with the present invention to carry out a typical task, such as programming a VCR.

[0060] 1. The user enters information concerning the desired action into the Remote Device via the Remote Client

[0061] 2. The Remote Device stores the information

[0062] 3. The user synchronizes the Remote Device by indicating to the Remote Client that the information should be transmitted

[0063] 4. The Remote Device dials into the Personal Server via cellular modem

[0064] 5. The Personal Server's Input/Output Module receives the phone call

[0065] 6. The Input/Output Module uploads the information from the Remote Device, creates a Message, and alerts the Scheduler/Router

[0066] 7. The Scheduler/Router determines that the Message is intended for the VCR Action Module

[0067] 8. The Scheduler/Router passes the message to the VCR Action Module, which parses the Message and in turn sends appropriate instructions to the VCR

[0068] 9. The VCR Action Module sends a new Message to the Scheduler/Router, confirming that the action was or was not taken, among other status details

[0069] 10. The Scheduler/Router logs, processes and passes the new Message to the appropriate Input/Output Module

[0070] 11. The Input/Output Module responds to the Remote Device, if necessary, reestablishing the connection if necessary

[0071] 12. The Remote Device displays relevant status information to the user via the Remote Client

[0072] 13. The Input/Output Module hangs up the modem connection as necessary

[0073] b. Network-Mediated Connection.

[0074] The following flowchart illustrates, as an alternate embodiment, the method of using a device constructed in accordance with the present invention to carry out a typical task using the Internet as an intermediary communications mechanism. The user accesses and logs onto the Service using the Remote Client running on the Remote Device.

[0075] 1. The Service presents the Remote Client with a Web page designed as an interface for programming a VCR

[0076] 2. The user enters the appropriate information and indicates that the data is complete

[0077] 3. The Service dials into the Personal Server via dial-up or other connectivity

[0078] 4. The Personal Server Input/Output Module receives the call

[0079] 5. The Input/Output Module uploads the information from the Service, creates a Message, and alerts the Scheduler/Router

[0080] 6. The Scheduler/Router determines that the Message is intended for the VCR Action Module

[0081] 7. The Scheduler/Router passes the message to the VCR Action Module, which in turn parses the message and sends appropriate instructions to the VCR

[0082] 8. The VCR Action Module sends a new Message to the Scheduler/Router, confirming that the action was or was not taken, among other status details

[0083] 9. The Scheduler/Router logs, processes and passes the new Message to the appropriate Input/Output Module

[0084] 10. The Input/Output Module responds to the Service, if necessary, reestablishing the connection if need be.

[0085] 11. The Service creates a Web page displaying relevant status information to the user via the Remote Client

[0086] 12. The Input/Output Module closes the connection to the Service.

[0087] Either of the above flowchart embodiments may be applied, with modifications, to the control and monitoring of objects other than the VCR, and to other system embodiments described herein.

[0088] 3. Functionality

[0089] The Personal Server is designed to carry out three functions, among others: control, monitoring and remote information tasks. Other functions are obvious to one of ordinary skill in the art. The Personal Server is typically used to control and monitor the following types of Controlled Devices: remote-ready objects, non-remote-ready objects and other objects. Many Controlled Devices will have both control and monitoring aspects to them, (e.g. "is the porch light on?""turn on the porch light"), though some will have relatively more of one type of functionality than the other. As an example, VCR's have relatively more control functions, relating to programming the VCR, than monitoring/status functions.

[0090] Typically, within the home or office, the Personal Server and its Controlled Devices will operate on a wide area network ("WAN") or local area network ("LAN"). In a preferred embodiment, Intel's BLUETOOTH™ is the hardware standard and protocol used to put together the network. Many other hardware and protocol implementation are obvious to one of ordinary skill in the art. In general, communication nodes will be used to broadcast the network signals to Controlled Devices on the network. For example, in one embodiment, X10 stations are used with the present invention to broadcast the signals.

[0091] a. Remote-Ready Objects

[0092] Remote ready Controlled Devices are appliances that are already remote-capable. These objects typically include VCRs, TVs, CD players, home or office security systems, and other sophisticated electronic devices that normally come with remote capability (generally using infra-red signals, in the current art). In addition, there are many standard household controls such as light switches, thermostats, garage doors, and alarm systems that are designed specifically for home-automation purposes. The Personal Server takes advantage of such remote capability to communicate with these devices. Many Controlled Devices use standardized communication protocols, which makes it a straightforward matter to communicate with these devices ("universal" remotes, for instance, take advantage of these standards). The Personal Server can be programmed with additional Input/Output Modules to allow for communication with non-standard objects, however. Input/Output Modules may be developed by value-added providers to enable the Personal Server to communicate with new and non-standard devices as they are developed.

[0093] As a further illustration, consider the activity of programming a VCR, discussed in the above section on overall architecture. The user, could, of course, program the VCR directly via the VCR console or remote. The present invention makes it a simple matter to program the VCR from the computer that runs the Personal Server. The user will typically enter the time and channel to record, or else a code corresponding to a program (such as a VCR-PLUS™ code). In a preferred embodiment, the user is also able to enter the name of the program, and the Personal Server, by interacting with a database or data source (such as a database available on the Internet), determines the program specifics. The Personal Server is sophisticated enough in its architecture to prompt the user if there is problem with the information entered, or if it cannot complete the task (for instance, if the VCR is already programmed for another program at the same time). It will also prompt the user with other status information, when it is appropriate.

[0094] Of course, the user generally will wish to program the VCR from a Remote Device rather than from the Personal Server itself. The present invention, by connecting the Remote Device to the Personal Server in a seamless fashion, makes this effectively the same task.

[0095] b. Non-Remote-Ready-Objects

[0096] Non-remote-ready Controlled Devices are those objects that typically are not remote capable. Examples of these objects include microwave ovens, dishwashers, toasters and coffee makers. Increasingly, such devices are being manufactured remote-ready. As Personal Servers become increasingly common, this trend will likely continue. For objects that are not remote-ready, a user will be able to adapt the objects for remote use with additional hardware. At the vely least, such objects can be controlled with simple commands by installing remote switches such as X10™ units (see "Other objects," below), or, failing that, at least simple on/off switches.

[0097] The programming of a non-remote-ready device is similar in implementation to the programming of a VCR outlined above One difference though is that non-remote-ready objects tend to be more dependent on status in order to function in an appropriate manner For instance, there should be coffee in the coffee maker or food in the microwave oven before the Personal Server activates these objects. It is partially for this reason that such objects have not been as readily adapted for remote use as some others have. Leaving a tape in a VCR and then wishing to program it later is a common desire. Leaving dirty clothes in a washing machine and washing them later is not so common. Nonetheless, the ability to do so must be convenient in some cases, such as turning a coffee machine on in the morning. As Personal Servers become more common, users will wish to take advantage of these conveniences, and thus more objects not envisioned as readily adaptable to remote use will be made remote-ready.

[0098] c. Other objects

[0099] There are a number of other objects that can be controlled and monitored with the Personal Server. For example, simple objects such as lighting fixtures can be equipped with X-10™ control units, which can be used to turn them on and off and to dim them. Much more sophisticated objects, such as pools and Jacuzzis, environmental systems, weather stations and television cameras, among others, can be controlled and monitored with the present invention. Again, the user may well need to adapt these objects for use with the Personal Server by installing hardware attachments.

[0100] One form of Controlled Device that merits special attention is a home or office computer. Either the Personal Server itself, or a separate computer, may function as a Controlled Device when operated in connection with the present invention, operated remotely via the Remote Client to perform a variety of tasks such as sending or retrieving electronic mail, voice mail, or faxes, uploading and downloading files, and connecting to the Internet.

[0101] The types of Controlled Devices that can be incorporated into the Personal Server system are almost limitless.

As one example, the system can be used to detect how many cars are sitting in the garage or driveway through the use of cameras, external sensors or chips embedded in cars. The latter is a particular cheap and simple way of bringing automobiles into the domain of the Personal Server. More sophisticated control features, such as remote car warmers, security systems or ignition devices, will become amenable to the present invention as available technology improves, and as users, vendors and inventors become more accustomed to and imaginative about such uses. One of ordinary skill in the art can imagine boundless examples. In this way, the present invention provides a broad basis for future technical development.

[0102] d. Remote Information Tasks

[0103] One of ordinary skill in the art will appreciate that remote information uses will also proliferate as technology, commercial innovation and commercial imagination develop. One current use is the transfer of computer files, such as video, spreadsheets, word processing documents and figures between the Remote Client and the Personal Server. These files may be used as part of the various control and monitoring features of the Personal Server, for example, remote viewing of images or streaming video from household cameras, or they may be entirely unrelated.

[0104] Communication can be done continuously, or in bursts, depending on need. Either the Remote Client of the Personal Server, and in some embodiments, objects in the network, can initiate and terminate communications. If there is a calculation or process that takes a great deal of time, the user may initiate the process remotely, terminate communication, and then check in from time to time to see if the process or calculation has been completed.

[0105] In one embodiment, the Personal Server can act as a pass-through communications link for the Remote Client. For instance, the user can surf the Internet remotely from the Remote Device via the Personal Server. Computational tasks and file retrieval can be done in a similar manner. The user can accomplish these tasks in real-time or else send the task to the Personal Server and then end the transmission. At some later time, when the Personal Server has completed the task or requires additional information, the Personal Server may request that communication be reestablished.

[0106] One particularly convenient use for the present invention applies to credit-card transactions. Merchants using the current invention can verify credit-card numbers by uploading them from the Remote Device (which will generally have a card reader) to the Personal Server for verification. A credit-card charge can be carried out in a similar manner. Other, transactions, financial and otherwise, are obvious to one of ordinary skill in the art.

[0107] 4. Firewall Penetration

[0108] In one embodiment of the present invention, the Personal Server network system is adapted to operate with protected networks. For this embodiment, the Personal Server and Controlled Devices, illustrated in **FIG. 2**, are coupled over a WAN, typically the Internet. The Personal Server is protected by a network protection or security system. Such a protection mechanism is typified by a firewall that shields one network from another network (e.g., the Internet), by blocking unwanted input to the internal network. Because they provide blocking and protection functions, firewalls, proxy servers, and other types of protection schemes are all impediments to making a TCP/IP or UDP connection to a computer from a remote device. To allow devices to access computers and other resources behind a firewall, the communication system must be configured to allow the firewall to permit certain types of communication to pass through it, while still maintaining its blocking function. Embodiments of the present invention provide means to identify the presence and type of firewall and then establish communications between the Personal Server and the Controlled Devices through the firewall mechanisms.

[0109] **FIG. 8A** illustrates a Personal Server network that includes a firewall detection and penetration scheme, according to one embodiment of the present invention. In system **800**, Personal Server **803** is coupled to the Internet **805** (or other WAN) through firewall **801**. Firewall **801** may be implemented as a single router or a combination of routers and server computers that perform firewall protection functions. A Connection Server **804** resides on the Internet **805**. The Connection Server **805** is a trusted server that is coupled to a variety of remote devices **806-812** through direct or indirect wireless access. These remote devices may be wireless devices, such as cell phones **806**, PDA devices **808**, wireless computers **810**, and the like, which transmit and receive data signals via transmission tower **816** through a wireless gateway **814** to the Internet **805** over wireless links. The remote devices illustrated in **FIG. 8A** may be Internet-enabled devices that connect to the Internet using their own internal Internet browsing abilities, such as a web browser on a laptop computer **810**. Other remote devices, such as cell phone **810**, may be Wireless Application Protocol (WAP) devices, or PDA devices that include built-in browser capabilities. Other remote devices include web kiosks, and WebTV systems, and the like. The remote devices may also include devices that communicate directly with the Personal Server **803** over the Internet using TCP/IP, without using a web-based interface.

[0110] The Connection Server **804** establishes a connection between the Personal Server **803** and the remote devices **806-812**. In a web-based embodiment, the Connection Server **804** presents correctly formatted web pages to the remote devices and uses information from the web pages to send commands to the Personal Server **803** and to present new web pages to Internet-enabled remote devices based on information from the Personal Server. Thus, the Connection Server **804** provides web-serving functions that allow a remote device user to access the Personal Server over the Internet. Firewall **801** protects the Personal Server **803** against unwanted access from the Internet, and keeps the internal network segments secure, for example between Personal Server **803** and locally networked file server **802**. For the sake of terminology, the Personal Server **803** and file server **802** network is considered to be "inside" the firewall **801**.

[0111] In general, the Personal Server **803** is coupled to the Internet **805** through a TCP/IP (Transmission Control Protocol/Internet Protocol) network connection. In an IP network, each computer is allocated a unique IP address. In a TCP/IP network, an IP address is usually shown in the form of an IP Address and a Port. The IP Address is a "dot" number (e.g., 123.333.5.20) and the port is a number in the range of 0 to 65,5535. Generally a computer or network element will have a single IP address and up to 64K ports. An IP Address/Port pair may be used to establish an outgoing connection from the computer, and it may be used to listen for and establish an incoming connection.

[0112] Many ports are used for standard communication functions. For instance, Port **80** is typically used to send and retrieve standard Web pages; and Port **443** is typically used to send and retrieve secure Web pages. Because there are so many ports and because different programs and applications may use these ports for different types of communications, leaving an IP address open to the Internet may leave it open to an unwanted or malicious communication from the outside. The purpose of a firewall is to impede these unwanted communications. Thus, firewall **801** in **FIG. 8A** acts to limit the type and range of connections to and from the user computer **804**.

[0113] As illustrated in **FIG. 8**A, Personal Server **803** includes a client application, referred to as a "connection module"**818** that establishes a connection from inside the firewall to the Connection Server **804**, and then keeps the connection open as a continuing communication conduit. The Communication Server **804** may have a corresponding "bridge module" (not shown) that transmits and receives data to the connection module **818**.

[0114] Some firewalls prevent certain types of information packets, such as UDP (User Datagram Protocol) packets, from going in or going out. UDP, along with TCP is a transport protocol within TCP/IP. While TCP ensures that a message is sent accurately and in its entirety, UDP does not provide robust error correction mechanisms, and is used for data, such as real-time voice and video, where there is limited time or reason to correct errors. In one embodiment of the present invention, the system packages these packets into an allowed data stream, such as TCP/IP, and then unpacks the stream at the other end of the communication conduit. If packets are destined for blocked ports, these packets are redirected through the conduit and then sent to the correct port when they reach the other side.

[0115] Various different types of firewalls and protection mechanisms exist. The different classes of firewalls described are IP Filtering, Network Address Translation, Proxy Servers, Stateful Firewalls, and Dynamic IP Addresses, and each poses an impediment to connectivity. The firewall penetration mechanism of the present invention can work with each type of firewall individually or any combination of these firewalls.

[0116] Because different firewalls and different proxy servers use a combination of different protocols, the firewall penetration system includes processes that determine what protocols are being used and to dynamically connect the Personal Server to the wireless network served by the Connection Server and configure the messages accordingly. To do this, upon installation, a process on the Personal Server establishes communication with the Communication Server, announces its presence and requests that the Communication Server begin a series of tests to try to connect back to the Personal Server. A series of tests is then run using communication protocols of increasing complexity until one is found that works. The Personal Server and the Connection Server then record that as the preferred method of communication between the two. The connection module **818** on the Personal Server then uses the preferred protocol to establish a connection to the Communication Server. This method thus determines whether a firewall **801** exists between the Personal Server and the Internet, and the type of firewall that exists Firewall penetration is accomplished because it is the computer on the inside of the firewall, i.e., Personal Server **803**, that initiates the connection. When the Personal Server creates a connection to the Connection Server, it announces its location (IP address), and updates its location every time it changes. In creating the connection from inside the firewall, the Personal Server formats the information using a format and protocol that the firewall will recognize and allow to pass through.

[0117] The different connection configurations in the order of increasing complexity that the connection module **818** attempts to connect to the Connection Server **804** are listed as follows:

[0118] 1. No firewall or proxy server

[0119] 2. Fixed IP Address (IP Filtering)

[0120] 3. Dynamic IP Address

[0121] 4. Network Address Translation Firewall

[0122] 5. Proxy Server

[0123] 6. Complex or Stateful Firewall

[0124] The processes executed by the connection module and Connection Server in establishing communication through each of these types of firewalls is provided in the description below.

[0125] a. IP Filtering

[0126] In an IP Filtering type of firewall, only certain port addresses are allowed to connect to the Internet. Usually these are port **80**, for standard web page access; and port **443** for SSL (Secure Sockets Layer) and secure web page access. For this type of firewall, the Connection Server is set to listen on port **443**. Thus, when the connection module of the Personal Server establishes a connection to the Connection Server, it does so over an allowed port. This is an "on-demand" type of connection in which the connection between the Connection Server **804** and the Personal Server **803** is opened only when there is data to be transmitted.

[0127] b. Dynamic IP Addresses

[0128] For dynamic IP address protection schemes, IP addresses of the connecting computer are changed with each access. That is, every time the connecting computer is given access to the Internet, it is assigned a new IP Address/Port pair, thus making it difficult to consistently locate.

[0129] For this type of connection, when the Personal Server obtains an Internet connection, the connection module registers its new IP address with the Connection Server, which logs it and uses it for subsequent connections. This way the Connection Server acts like a directory service for an outside application trying to establish an inbound connection to the user computer. Like the IP filtering system the dynamic IP address system is an on-demand system.

[0130] c. Network Address Translation (NAT) Firewalls

[0131] In a Network Address Translation type of firewall, each IP Address/Port pair on the computer behind the firewall is translated to a different IP Address/Port pair. This enables a local area network to use one set of IP addresses for internal traffic and a second set of addresses for external traffic. A NAT device located where the LAN meets the Internet makes all necessary IP address translations.

[0132] For this type of firewall, like the dynamic IP address solution, the connection module of the Personal Server registers its new address with the Connection Server. If the communication between the Personal Server and the Connection Server breaks, the Personal Server reconnects. Communication through a NAT firewall is also on-demand.

[0133] d. Proxy Servers (SOCKS 4 Proxy, SOCKS 5 Proxy, HTTP Proxy)

[0134] A proxy is a device that acts on behalf of another device. For web applications, a web proxy acts as a partial web server, in which a network client makes requests to the proxy, which then makes requests on their behalf to the appropriate web server. Proxy servers allow many computers to access the Internet through a single Internet connection, which is done by temporarily assigning a port of the Internet connection to the user computer. Unlike NAT and dynamic IP address schemes, web proxying is not a transparent operation, and must be explicitly supported by the clients. For this type of firewall, each IP Address/Port pair on the computer behind the firewall is translated to a different IP Address/Port pair. Inbound connections and UDP connections are not allowed. Only outgoing TCP/IP connections to port **80** and port **443** are allowed.

[0135] To penetrate this proxy server firewall, the Connection Server listens on port **443**, the port normally used for secure web pages. The connection module of the Personal Server establishes a TCP/IP link to the Connection Server on port **443** and keeps the connection open by sending periodic bursts of data, referred to "keep alives." If the connection is broken, the connection module opens it again. On the Connection Server side, all incoming data is packaged into a single TCP/IP stream that is sent over the conduit established by the connection module. The connection module unpacks the data on the client side, and sends the information to the appropriate ports on the Personal Server (the computer on which it is running). When the Personal Server sends information back to the Connection Server, it packages it in the same way, sends it over the conduit. The Personal Server then unpacks the data stream to send to the remote devices. At installation, the Personal Server first attempts Socks **5**, then Socks **4**, and then HTTP-proxy protocol.

[0136] e. Stateful Firewalls

[0137] A normal Firewall is "stateless" because it has no memory of context for connection states, and each connection through it is a new connection. A stateful firewall remembers the context of connections and continuously updates this state information in dynamic connection tables. This type of firewall monitors the information flowing through it and only allows certain types of data in certain states to pass through. Thus, if a foreign packet tries to enter the network, claiming to be part of an existing connection, the firewall can consult the connection tables. If a packet does not match any of the established connections, that packet is dropped. For example, a stateful firewall can monitor web transactions for proper HTTP formatting and proper HTTP responses. It then allows only connections of short duration, such as a web page access.

[0138] For stateful firewalls, the Connection Server is set to listen on port **443** (the HTTP port). This is the secure port for web page access, so that the firewall will not filter out its IP address. Since data that passes through this port is normally encrypted, the firewall allows all information through and cannot monitor its state. When the connection is broken by the statefil firewall, the connection module automatically re-establishes a connection to the Connection Server and keeps the connection alive as long as it can by sending periodic bursts of data, "keep alives."

[0139] Once communication has been established between the Personal Server and the Connection Server through the firewall, the remote devices can be used to access the Personal Server. In one embodiment, a remote device **806** transmits a login request to the Connection Server **804** via the wireless service **814** The Connection Server **804** authenticates the login, and sends a request to the Personal Server **803**. The Personal Server then responds to the request, which is relayed through the Connection Server **804** to the remote device **806**. At this point, the remote device, using the conduit through the Connection Server **804**, has remote access and control to the Personal Server, and any resources coupled and controlled to the Personal Server, such as file server **802**, and any other desktop computers or devices.

[0140] The embodiment illustrated in **FIG. 8A** illustrates a configuration in which the Connection Server **804** resides on the Internet. Such a configuration may be used in an Application Service Provider (ASP) scenario in which the Connection Server **804** is hosted by an ASP or other third-party entity. In an alternative embodiment of the present invention, the Connection Server **804** may be hosted in-house, that is on the same protected network as the Personal Server **803**. Such a configuration, according to this alternative embodiment is illustrated in **FIG. 8B**. As shown in **FIG. 8B**, the remote devices **806-812** are coupled through the Internet **805** to a firewall protected network comprising Personal Server **803**, Connection Server **804**, and other resources, such as file server **802**. For this configuration, the Personal Server **803** establishes communication with the Connection server **804** through connection module **818** directly over the internal LAN link. For example, upon boot-up, the Personal Server can register with the Connection Server, which is hosted by the same entity, thereby opening a communication channel. The remote devices **806-812** transmit login requests to the Connection Server **804**, which authenticates the request and relays the request to the Personal Server **803**.

[0141] f. Method

[0142] **FIG. 9** is a flowchart that illustrates the method of identifying the presence of a firewall and establishing a communication conduit between a user computer and Personal Server, according to one embodiment of the present invention. The flowchart of **FIG. 9** illustrates the general process steps executed by the Personal Server and Connection Server for the network illustrated in **FIG. 8A** to detect and circumvent the various types of firewalls described above. In step **902**, the connection module in the Personal Server detects whether a firewall exists between it and the Connection Server by comparing the IP address of the machine on which the Personal Server is running to the IP address from which the connection was received. If such a firewall exists, the type of firewall is determined, step **903**.

In general, the types of connections to be established through any detected firewall fall into two general categories: on-demand connections **906**, and Personal Server initiated connections **910**.

[**0143**] On-demand protection connections **906** include IP filtering, dynamic IP addresses, and NAT firewalls that allow incoming connections. For these types of firewalls, the Personal Server attempts to establish a connection to the Connection Server so that the wireless remote devices coupled to the Connection Server can communicate with the Personal Server at will. The connection is initiated by the Connection Server and opened only when there is data to be transmitted between the two servers. The Connection Server listens on a secure port, typically port **443** for secure web page access, step **912**. The Personal Server then establishes a connection with the Connection Server over this secure port, step **914**. For this embodiment, it is generally assumed that dynamic IP addressing is used. In step **916**, the Personal Server registers its IP address with the Connection Server, and then waits for incoming connections from the Connection Server, step **918** If the connection is broken, as determined in step **920**, the Personal Server registers its address with the Connection Server again from step **916**. In this manner, the Connection Server can always establish a connection to the Personal Server even if the Personal Server has a dynamic IP address.

[**0144**] Personal Server initiated connections **910** are used for proxy servers, stateful fireballs, and NAT firewalls that refuse incoming connections. For Personal Server initiated connections **910**, the Connection Server listens on a secure port, e.g., port **443**, step **922**. The Personal Server then establishes a connection with the Connection Server over this secure port, step **924**. The firewall may cause connections to be repeatedly broken between the Personal Server and the Connection Server since it cannot monitor the state of any encrypted data that is transmitted. In step **928**, the process determines if the connection has been broken. If so, the Personal Server re-establishes the connection with the Connection Server, from step **924**. The Personal Server then maintains the connection to the Connection Server through periodic "keep alive" signals, step **926**.

[**0145**] Embodiments of the present invention may be used in conjunction with various encryption and authentication mechanisms to provide further security measures. For example, transmitted data may be encrypted using public key/private key and/or Secure Socket Layer (SSL) algorithms.

[**0146**] Although embodiments of the present invention have been described in relation to particular types of firewalls, it should be noted that the firewall penetration solutions described herein can be implemented with other types of firewalls that feature similar protection mechanisms.

[**0147**] In the foregoing, a system has been described for providing firewall penetration between two networks through a connection server Although the present invention has been described with reference to specific exemplary embodiments, it will be evident that various modifications and changes may be made to these embodiments without departing from the broader spirit and scope of the invention as set forth in the claims. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense. All publications and patents herein are incorporated by reference in their entirety.

What is claimed is:

1. A method of interfacing a user computer with a network comprising one or more client computing devices coupled to a server computer, the method comprising:

transmitting a test command from the user computer to the server computer to cause the server computer to transmit a return signal to the user computer to determine whether a firewall exists between the user and server computers;

transmitting a series of messages between the user computer and the server computer using communication protocols of increasing complexity to identify the type of firewall that exists, if it is determined that a firewall exists between the user and server computers;

utilizing the communication protocol corresponding to the type of firewall identified for communications between the user computer and the server computer; and

registering a network address of the user computer with the server computer if the firewall causes the address of the user computer to change upon each new connection with the server computer.

2. The method of claim 1 further comprising the steps of:

re-establishing communication from the user computer to the server computer by the user computer if the communication is unintentionally broken; and

maintaining the communication between the user computer and server computer by transmitting periodic non-traffic related signals from the user computer to the server computer.

3. The method of claim 2 wherein the communications protocols include, in order of increasing complexity: fixed address firewall, dynamic address firewall, proxy server protection, network address translation firewall, and stateful firewall.

4. The method of claim 3 wherein the network comprises a wireless network coupling one or more wireless client computing devices to the server computer.

5. The method of claim 4 wherein the one or more wireless client computing devices comprises one of: a personal computer, handheld personal digital assistant, and networkable cellular phone.

6. The method of claim 5, wherein the network comprises a TCP/IP network and the data transmitted over the network comprises one of: computer text data, audio data, and video data.

7. The method of claim 6 wherein the user computer and server computer are coupled through a bidirectional communications network that comprises the Internet.

8. The method of claim 7 wherein the server computer is coupled to the one or more wireless client computing devices over a remote control protocol.

9. The method of claim 8 wherein the server computer provides control and monitoring functionality over the one or more wireless client computing devices using a protocol comprising one of: TCP/IP protocol, **X10** protocol, and Bluetooth protocol.

10. A system comprising:

a first computer coupled to a network coupling one or more client computers;

a second computer including a connection module for communicating with the first computer;

a firewall protection mechanism disposed between the first computer and the second computer to prevent unwanted network access from the first computer to the second computer;

wherein the connection module is configured to initiate transmission of a series of messages between the first computer and the second computer using communication protocols of increasing complexity to identify the type of firewall that exists, and further configured to register an address of the first computer with the second computer if the firewall causes the address of the first computer to change upon each new connection with the second computer.

11. The system of claim 10 wherein the connection module is further configured to re-establish communication from the first computer to the second computer by the first computer if the communication is unintentionally broken, and maintain the communication between the first computer and second computer by transmitting periodic non-traffic related signals from the first computer to the second computer.

12. The system of claim 11 wherein the communications protocols include, in order of increasing complexity: fixed address firewall, dynamic address firewall, proxy server protection, network address translation firewall, and stateful firewall.

13. The system of claim 12 wherein the network comprises a wireless network coupling the one or more wireless client computing devices to the first computer.

14. The system of claim 13 wherein the one or more wireless client computing devices comprises one of: a personal computer, a handheld personal digital assistant, and a networkable cellular phone.

15. The system of claim 14, wherein the network comprises a TCP/IP network and the data transmitted over the network comprises one of: computer text data, audio data, and video data.

16. The system of claim 15 wherein first computer and second computer are coupled through a bi-directional communications network that comprises the Internet.

17. The system of claim 16 wherein the second computer comprises a server computer coupled to the one or more wireless client computing devices and communicating over a remote control protocol.

18. The system of claim 17 wherein the server computer provides control and monitoring functionality over the one or more wireless client computing devices using a protocol comprising one of: TCP/IP protocol, X10 protocol, and Bluetooth protocol.

19. A method for interfacing a first server computer to a second server computer through a network connection including a network firewall, the method comprising the steps of:

determining if the connection between the first server computer and the second server computer is initiated by the first server computer or by the second server computer;

causing the first server computer to listen for a connection to the second server computer over a secure port accessible by the first server computer;

establishing a connection between the first server computer and the second server computer over the secure port;

registering a network address of the second server computer with the first server computer, if the connection between the first server computer and the second server computer is initiated by the first server computer; and

re-registering the network address of the second server computer with the first server computer if the connection established between the first server computer and the second server computer is broken.

20. The method of claim 19 wherein the firewall comprises one of: an address filtering firewall, a dynamic address firewall, and a network address translation firewall that allows incoming connections to the second server computer.

21. The method of claim 19, wherein if the connection between the first server computer and the second server computer is initiated by the second server computer, the method comprises the steps of:

causing the first server computer to listen for a connection to the second server computer over a secure port accessible by the first server computer;

establishing a connection between the first server computer and the second server computer over the secure port;

determining whether the connection has been broken;

re-establishing the connection between the first server computer and the second server computer; and

transmitting periodic non-data signals from the second server computer to the first server computer to maintain the connection.

22. The method of claim 21 wherein the firewall comprises one of: a proxy server firewall, a stateful firewall, and a network address translation firewall that refuses incoming connections to the second server computer.

23. The method of claim 21 wherein the first server computer is coupled to one or more remote computing devices over a wireless network link, and wherein the first server computer is coupled to one or more user computers over a Local Area Network link.

24. The method of claim 23 wherein the first server computer and second server computer are coupled through a bidirectional communications network that comprises the Internet.

25. The method of claim 16 wherein the first server computer comprises is remotely coupled to the plurality of the one or more remote computing devices over a remote control protocol, and provides control and monitoring functionality over the one or more remote computing devices using a protocol comprising one of: TCP/IP protocol, **X10** protocol, and Bluetooth protocol.

* * * * *

US006219786B1

(12) **United States Patent**
Cunningham et al.

(10) **Patent No.: US 6,219,786 B1**
(45) **Date of Patent: Apr. 17, 2001**

(54) **METHOD AND SYSTEM FOR MONITORING AND CONTROLLING NETWORK ACCESS**

(75) Inventors: **Mark Cunningham**, Leek; **Andrew Trevarrow**, Withington, both of (GB)

(73) Assignee: **SurfControl, Inc.**, Scotts Valley, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/150,264**

(22) Filed: **Sep. 9, 1998**

(51) **Int. Cl.**^{7} **G06F 13/00**

(52) **U.S. Cl.** **713/152**; 713/201; 713/153; 709/229

(58) **Field of Search** 713/200, 201, 713/202, 151, 152, 153; 709/229; 340/825.31, 825.34

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,720,033		2/1998	Deo	395/186
5,727,146		3/1998	Savoldi et al.	395/187.01
5,742,759		4/1998	Nessett et al.	395/187.01
5,826,014	*	10/1998	Coley et al.	713/201
5,835,722		11/1998	Bradshaw et al.	395/200.55
5,884,033		3/1999	Duvall et al.	395/200.36
5,889,958	*	3/1999	Willens	709/229
6,061,798	*	5/2000	Coley et al.	713/201

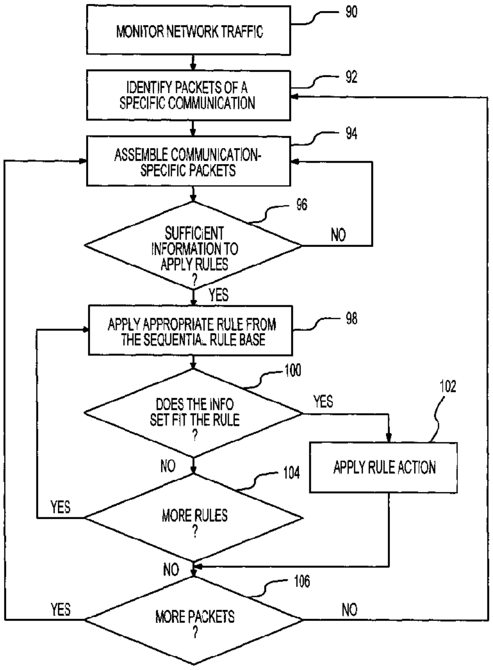
* cited by examiner

Primary Examiner—James P. Trammell
Assistant Examiner—Pierre Eddy Elisca
(74) *Attorney, Agent, or Firm*—Terry McHugh; Law Offices of Terry McHugh

(57) **ABSTRACT**

A method and system for monitoring and controlling network access includes non-intrusively monitoring network traffic and assembling data packets that are specific to individual node-to-node transmissions in order to manage network access both inside and outside of a network. A rules base is generated to apply at either or both of the connection time and the time subsequent to connection. With regard to a particular node-to-node transmission, the data packets are assembled to identify the source and destination nodes, as well as contextual information (i.e., ISO Layer 7 information). The access rules are applied in a sequential order to determine whether the transmission is a restricted transmission. The rules are maintained in a single rules base for the entire network and are distributed to each monitoring node. Any of the protocols in the suite of TCP/IP protocols can be managed. The result of an analysis against the rules base causes a connection attempt to be completed or denied, a previously established connection to be broken, logging to occur, or a combination of these and other actions. Data collected during connection attempts or during a connection's lifetime may be passed to a third-party hardware or software component in order for independent validation to take place. Traffic monitoring and access management can be executed at a node other than a choke point of the network.

18 Claims, 7 Drawing Sheets


MONITOR NETWORK TRAFFIC
90
IDENTIFY PACKETS OF A SPECIFIC COMMUNICATION
92
ASSEMBLE COMMUNICATION-SPECIFIC PACKETS
94
SUFFICIENT INFORMATION TO APPLY RULES ?
96
NO
YES
APPLY APPROPRIATE RULE FROM THE SEQUENTIAL RULE BASE
98
DOES THE INFO SET FIT THE RULE ?
100
YES
102
APPLY RULE ACTION
NO
104
YES
MORE RULES ?
NO
106
YES
MORE PACKETS ?
NO

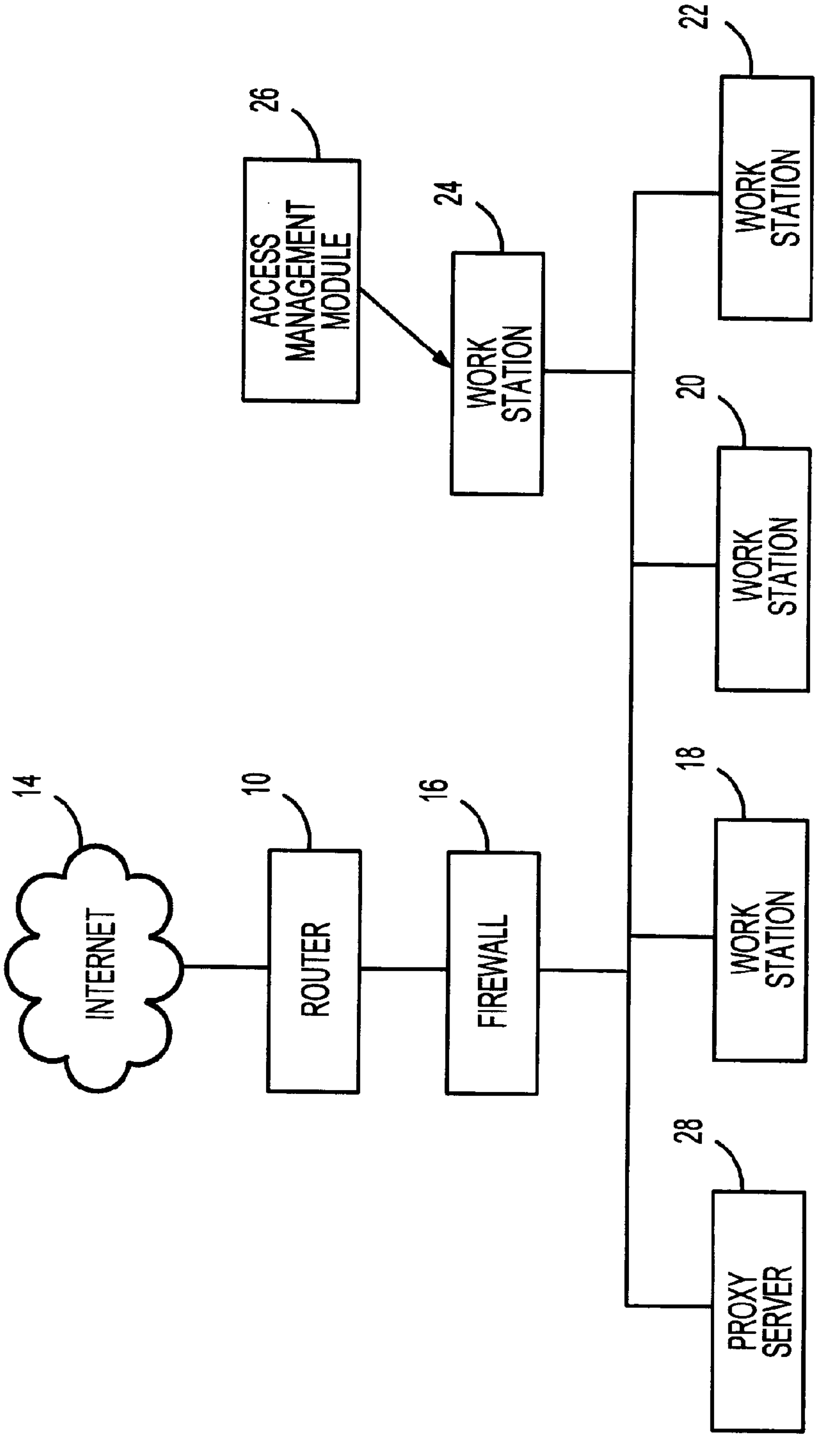

14
INTERNET
10
ROUTER
16
FIREWALL
26
ACCESS MANAGEMENT MODULE
24
WORK STATION
18
WORK STATION
20
WORK STATION
22
WORK STATION
28
PROXY SERVER

FIG. 1

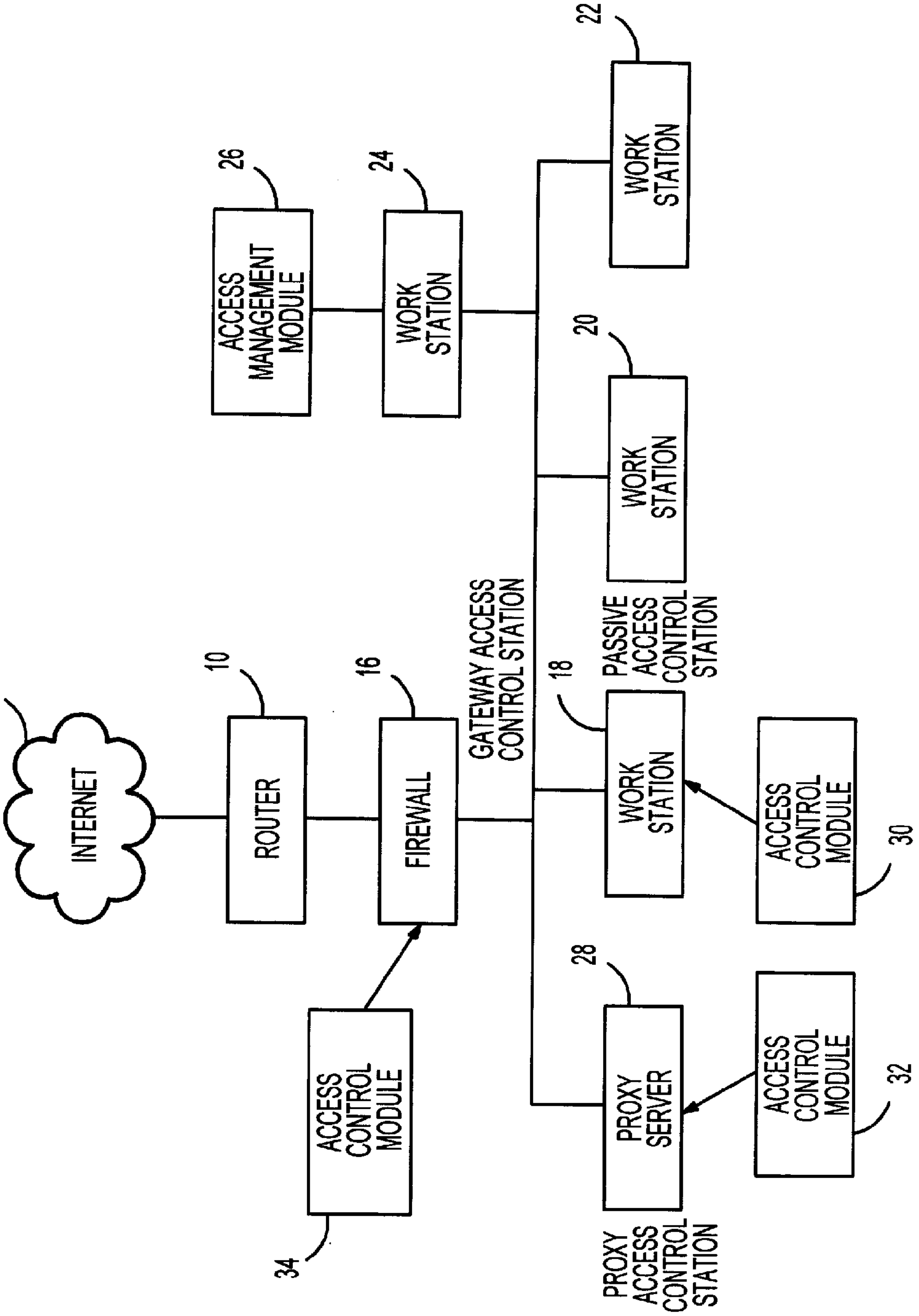

14
INTERNET
10
ROUTER
16
FIREWALL
ACCESS CONTROL MODULE
34
GATEWAY ACCESS CONTROL STATION
26
ACCESS MANAGEMENT MODULE
24
WORK STATION
18
WORK STATION
PASSIVE ACCESS CONTROL STATION
20
WORK STATION
22
WORK STATION
28
PROXY SERVER
PROXY ACCESS CONTROL STATION
ACCESS CONTROL MODULE
30
ACCESS CONTROL MODULE
32

FIG. 2

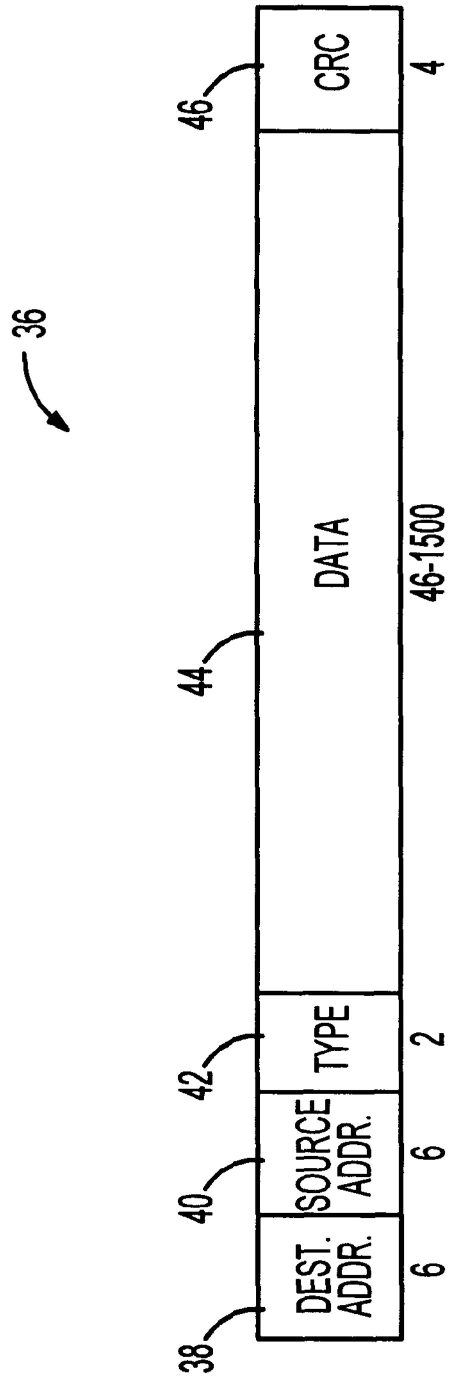


FIG. 3
(PRIOR ART)

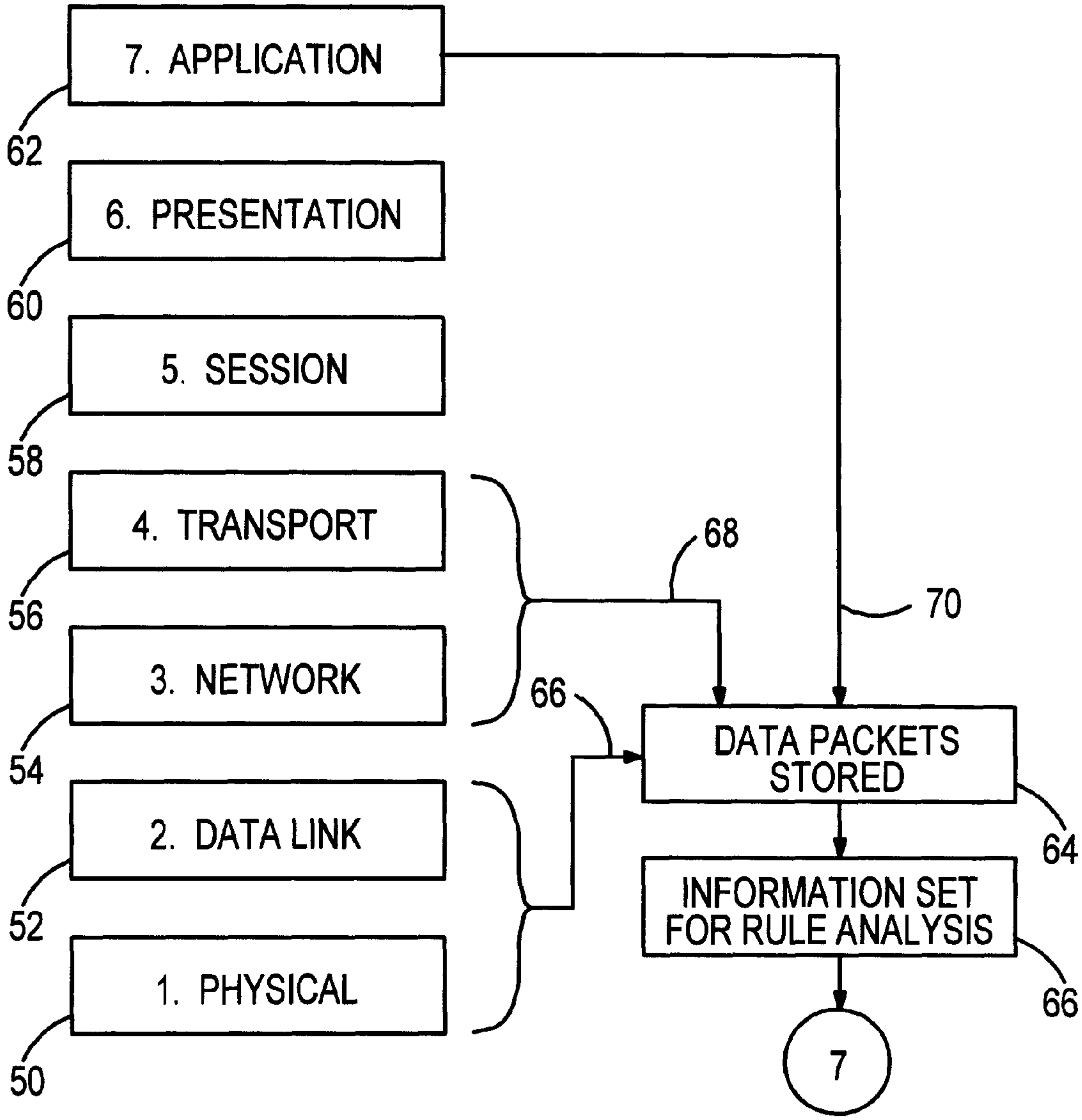
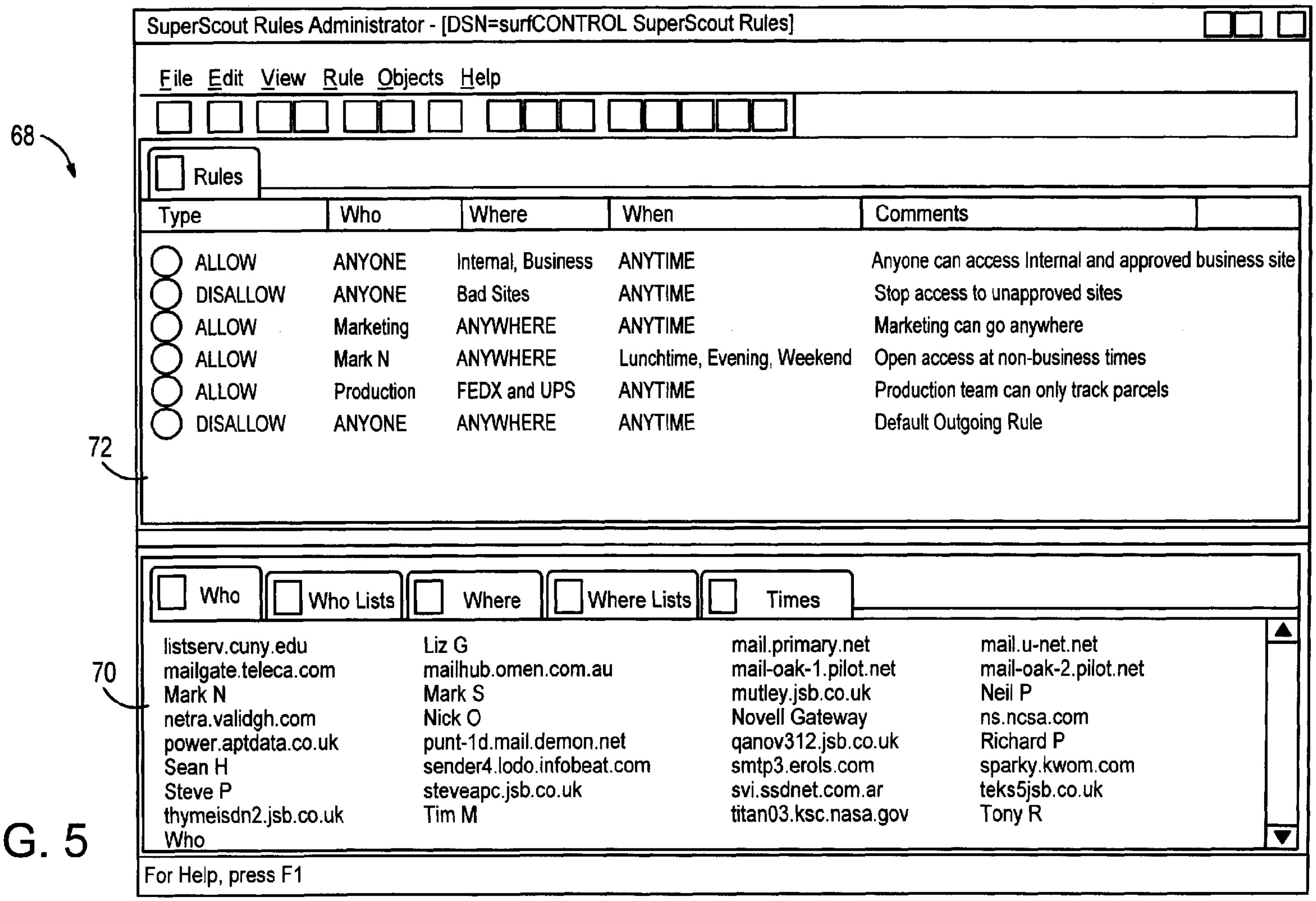

7. APPLICATION
62
6. PRESENTATION
60
5. SESSION
58
4. TRANSPORT
56
3. NETWORK
54
2. DATA LINK
52
1. PHYSICAL
50
68
70
66
DATA PACKETS STORED
64
INFORMATION SET FOR RULE ANALYSIS
66
7

FIG. 4


68
72
70
SuperScout Rules Administrator - [DSN=surfCONTROL SuperScout Rules]
File Edit View Rule Objects Help
Rules
Who
Who Lists
Where
Where Lists
Times
For Help, press F1

Type	Who	Where	When	Comments
○ ALLOW	ANYONE	Internal, Business	ANYTIME	Anyone can access Internal and approved business site
○ DISALLOW	ANYONE	Bad Sites	ANYTIME	Stop access to unapproved sites
○ ALLOW	Marketing	ANYWHERE	ANYTIME	Marketing can go anywhere
○ ALLOW	Mark N	ANYWHERE	Lunchtime, Evening, Weekend	Open access at non-business times
○ ALLOW	Production	FEDX and UPS	ANYTIME	Production team can only track parcels
○ DISALLOW	ANYONE	ANYWHERE	ANYTIME	Default Outgoing Rule

listserv.cuny.edu
mailgate.teleca.com
Mark N
netra.validgh.com
power.aptdata.co.uk
Sean H
Steve P
thymeisdn2.jsb.co.uk
Who

Liz G
mailhub.omen.com.au
Mark S
Nick O
punt-1d.mail.demon.net
sender4.lodo.infobeat.com
steveapc.jsb.co.uk
Tim M

mail.primary.net
mail-oak-1.pilot.net
mutley.jsb.co.uk
Novell Gateway
qanov312.jsb.co.uk
smtp3.erols.com
svi.ssdnet.com.ar
titan03.ksc.nasa.gov

mail.u-net.net
mail-oak-2.pilot.net
Neil P
ns.ncsa.com
Richard P
sparky.kwom.com
teks5jsb.co.uk
Tony R

FIG. 5

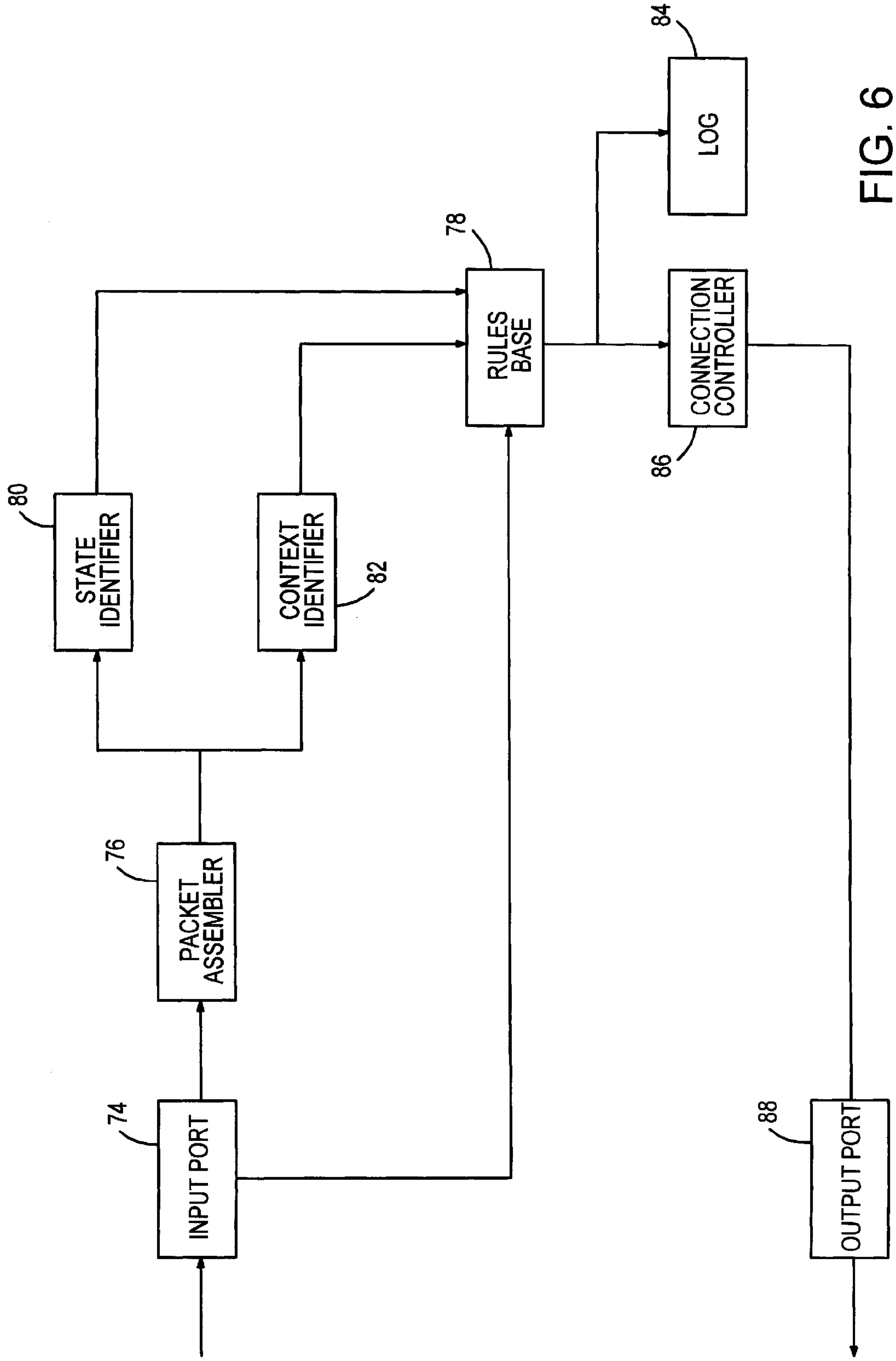

INPUT PORT
74
PACKET ASSEMBLER
76
STATE IDENTIFIER
80
CONTEXT IDENTIFIER
82
RULES BASE
78
LOG
84
CONNECTION CONTROLLER
86
OUTPUT PORT
88

FIG. 6

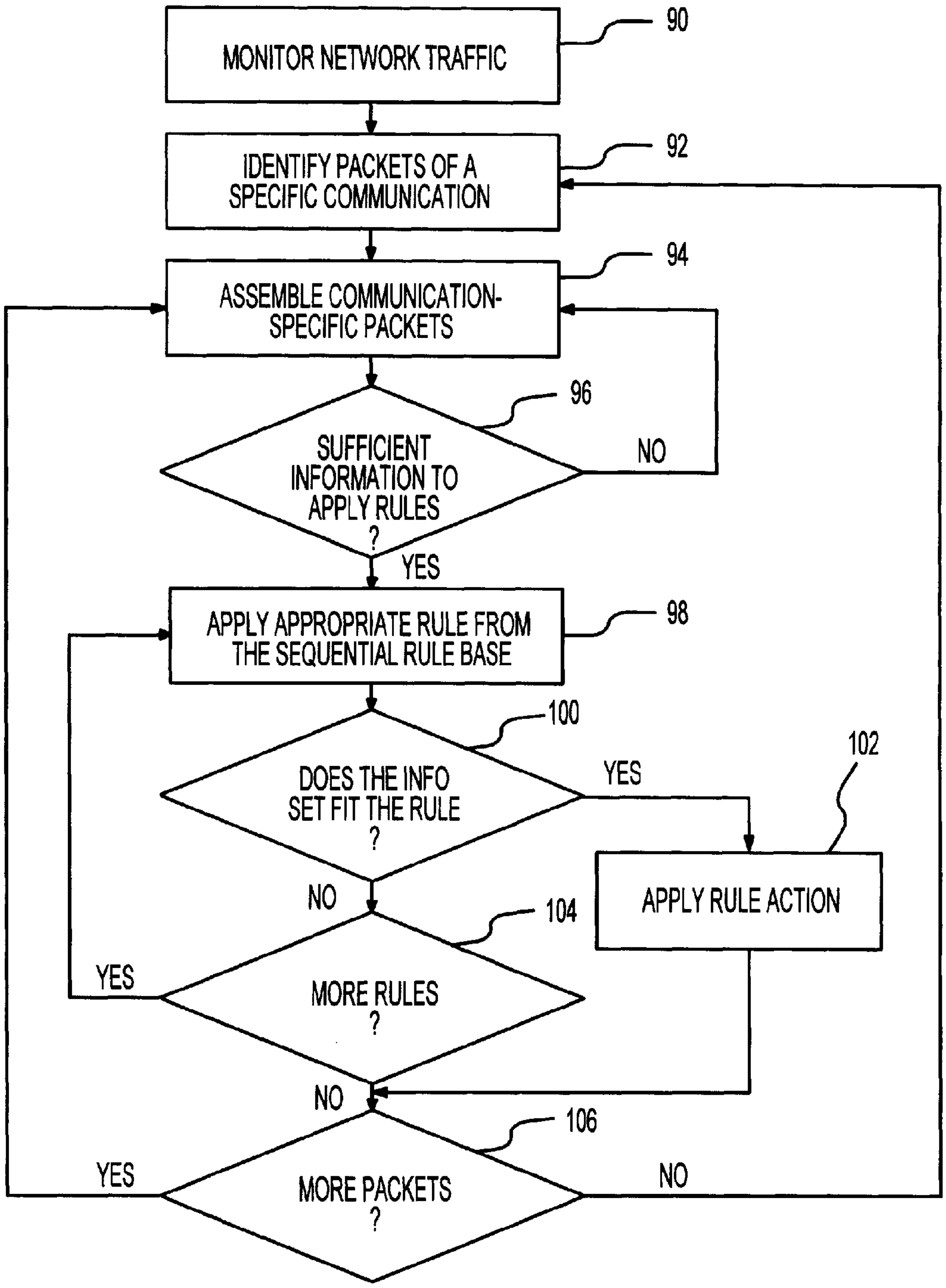

MONITOR NETWORK TRAFFIC
90
IDENTIFY PACKETS OF A SPECIFIC COMMUNICATION
92
ASSEMBLE COMMUNICATION-SPECIFIC PACKETS
94
SUFFICIENT INFORMATION TO APPLY RULES ?
96
NO
YES
APPLY APPROPRIATE RULE FROM THE SEQUENTIAL RULE BASE
98
DOES THE INFO SET FIT THE RULE ?
100
YES
102
APPLY RULE ACTION
NO
MORE RULES ?
104
YES
NO
MORE PACKETS ?
106
YES
NO

FIG. 7

METHOD AND SYSTEM FOR MONITORING AND CONTROLLING NETWORK ACCESS

TECHNICAL FIELD

The invention relates generally to a method and system for managing access control to resources of a distributed network, and relates more particularly to monitoring and controlling computer users' access to network resources from both inside and outside the network.

BACKGROUND ART

There are a number of available topologies for computer networks of nodes. A computer network may be highly centralized, having a mainframe computer that is accessed by a number of user computers, such as desktop computers. Currently, the trend is away from centralization and toward distributed processing and client-server relationships. In a distributed network, intelligence and processing power are distributed among a number of network nodes, typically with client workstations communicating with distributed servers. Other relationships among nodes of a network are known.

A network of nodes may be associated with a single enterprise, such as a local area network (LAN) of a particular business. Such a network enables communications and data exchanges among the various nodes of the network. A single protocol may be used in the accessing of resources within the LAN. Thus, when a first node, such as a client workstation, accesses the computing resources of a second node, such as a server for storing various applications, data is exchanged without requiring a protocol conversion.

However, the largest and most pervasive network is the nonproprietary global communications network referred to as the Internet. A number of different network protocols are used within the Internet. Protocols that fall within the Transmission Control Protocol/Internet Protocol (TCP/IP) suite include the HyperText Transfer Protocol (HTTP) that underlies communications via the World Wide Web, TELNET for allowing access to a remote computer, the File Transfer Protocol (FTP), and the Simple Mail Transfer Protocol (SMTP) to provide a uniform format for exchanging electronic mail, as well as a number of standardized or proprietary protocols for multimedia and broadcast services.

An implementation of these and other Internet protocols solely within an organization is often referred to as an Intranet, while the use of such protocols across a restricted set of Internet sites that are relevant to a particular organization is referred to as the organization's Extranet.

Much attention has been given to installing computer network gateways which focus on ensuring that potential intruders (sometimes referred to as "hackers") cannot gain illegal access via the Internet to an organization's computing resources on their Intranets. These gateways are "choke points," through which network traffic that is to be controlled must flow. Such "firewalls" are configured to allow any outbound connection or traffic to occur, but to restrict inbound traffic to specific services that are deemed to be non-threatening to the organization. Firewalls may also perform a limited amount of "packet filtering," which attempts to control traffic by reference to non-contextual, low-level network packets.

An issue that receives less attention is ensuring that the employees of an organization are appropriately managed. This management extends to accessing external computer resources and accessing internal computer resources. The management may be set forth in an access control policy of the organization. With respect to many aspects, the management is the converse of the problem that firewalls are intended to solve. While firewalls are focused on keeping intruders from gaining unwanted accesses, access control systems are focused on ensuring that insiders are managed according to the access control policy of the organization.

There are a number of motivations for implementing an access control policy within an organization. With regard to controlling external communications, two important reasons are maximizing employee productivity by ensuring that Internet access is used primarily for business purposes and maximizing the Internet-connection capability (i.e., bandwidth) of the organization, particularly during peak usage times. For example, using streaming audio and video services at peak times of the day in terms of the network traffic of an organization can seriously diminish productivity of other users within the organization who are attempting to perform tasks such as e-mail file transfers, terminal emulations, and network database inquiries.

Using traditional approaches, organizations apply stringent rules and sometimes overbearing management dicta in order to prevent key business usage of the Internet from being adversely affected by casual or inappropriate usage. The traditional approaches are typically administratively difficult to set up and maintain, as well as being difficult to scale from small organizations to large enterprises. Thus, some of the productivity gains are negated by management overhead.

One traditional approach to providing access control with regard to resource requests generated within a network is to leverage firewall technology and focus on the well-known packet filtering techniques. This typically requires a computer system to be installed as a router with at least two network interface cards and with no data packets being allowed to be forwarded from one interface card to the other without prior filtering. That is, firewall technology has been "turned around" to form some degree of protection. Rather than controlling outsiders attempting to access resources of the network, the techniques are used to control insiders attempting to access external resources. This approach may work well in some applications, but in others the approach is too simplistic and inflexible.

U.S. Pat. No. 5,727,146 to Savoldi et al. describes a method for securing network access to a network. All data packets that are transmitted via the network are monitored for authorized source addresses, rather than examining only the initial network connection packets. Thus, network access to a port is secured by monitoring the source address of each packet that is sent as a device tries to train to the port of the network. If the source address matches an authorized source address assigned to the port to which the device is attached, the device is allowed access to the system. However, if the device attempts to train with a source address different from the authorized source address, all packets sent by the device are denoted as errored packets to prevent them from being accepted by any other device in the network. By monitoring all packets, the system detects occurrences in which a device attempts to "disguise" itself by first training with an authorized source address and then sending a packet with an unauthorized source address.

Another approach to implementing network access control is to add third-party software modules into commercially available proxy server products. For example, software modules that are dedicated to attempting to control access may be added to a web proxy server. The disadvantages of this approach include the fact that only a small subset of Internet protocols is actually routed through a web proxy server. These protocols are typically restricted to browser-based FTP, Gopher and WWW protocols. This subset of protocols does not include the protocols used in the transfer of packets for e-mail, telnet, other file transfers, and streaming audio and video. Therefore, using web proxy servers as choke points allows only an incomplete level of control.

Another approach to attempting control access is to establish "blacklists" or "control lists" into proxy servers or into individual client workstations. This is a somewhat simplistic approach to meeting the needs of organizations and is often administratively burdensome to corporations, since the lists must be updated on a regular basis.

What is needed is a method and system for providing access control to resources of a network in a manner that is flexible, scalable and relatively easy to administer.

SUMMARY OF THE INVENTION

A method and system in accordance with the invention are configured to provide access control to resources of a network by collecting and assembling data packets of a specific transmission, so as to enable identification of information from raw data packets at the lowest level to application-level data at the top-most level. In terms of the standardized model referred to as the International Standards Organization (ISO) model, the data packets are assembled to determine not only the lower-layer information from the headers of the packets, but also the uppermost Application Layer (i.e., Layer **7**) contextual information. Access rules are then applied to determine whether the specific transmission is a restricted transmission.

In the preferred embodiment, the steps of receiving and assembling the data packets occur non-intrusively with respect to impact on traffic flow through the network. That is, the data packets are intercepted without impact on network performance, unless a restricted transmission is detected. Receiving and assembling the data packets may occur at a workstation or server that is dedicated to providing access control. For example, a free-standing workstation may be connected as a node to the network and may be switched to a promiscuous mode in order to receive all data packets transmitted to or from other nodes of the network. This allows the workstation to receive the fragments (i.e., data packets) of each access attempt from elsewhere on the network to either external destinations or other internal destinations. The fragments are pieced together to identify ISO Layer **7** information, as well as lower layer information. In an e-mail context, the Application Layer information of interest may include the information contained within the "to," "from" and "subject" lines of e-mail messages. In a web context, the Application Layer information of interest may include the text of the HTML pages.

By placing the dedicated workstation or server outside of the direct paths from source nodes to destination nodes, the impact on network traffic is minimal. However, the method and system may also be implemented by examination and management at a choke point, such as a proprietary proxy server, a firewall or other network node that is acting as a gateway between the network and an external network (e.g., the Internet). The examination and management at a choke point may take the form of a plug-in module for receiving, assembling and examining data packets in the manner described above. However, the examination of access attempts at the choke point will not provide the level of access control available by monitoring all traffic within the network, and may well impact network performance. Therefore, the system may include both access monitoring at the choke point and non-intrusive monitoring elsewhere on the network.

In the approach in which access is examined non-intrusively, the dedicated workstation or server may be configured as a "bare-bones" TCP/IP virtual machine to establish a capability of providing information extending from the lower layers of the ISO model to the Application Layer. There may be more than one dedicated workstation or server, particularly if the network is divided into segments. The access rules are preferably stored as a rules base, which may be centralized if there is more than one node that provides access management. Alternatively, the rules base is configured at a single site, but then automatically distributed to each access control point on the network.

The access control rules may apply at the time that a connection is established or may depend upon application protocol data following a successful connection. In the preferred embodiment, the rules are applied both at the time of connection and subsequent to the connection, as data packets are assembled. If a node-to-node transmission is determined to be a transmission that is restricted by the rules base, a connection attempt may be denied, a previously established connection may be broken, a simple logging may occur, or a combination of these actions may be implemented. Data collected during the connection attempts or during a connection's lifetime may be passed to third-party software in order for independent validation to occur. However, this is not critical.

The rules base is preferably divided into two sets of rules. The first set relates to access management requirements with regard to outgoing connection attempts, while the second set relates to internal connection attempts. The rules within each set may be layered in order to allow seemingly inconsistent rules to be included in a single rules base. For example, rules within a particular set may be applied sequentially, so that a specific rule application is accessed prior to a general rule application that contradicts the specific rule. The rules base is preferably configured in terms that are familiar to users, such as usernames, group names, workstation identifiers, destination addresses and URLs, services required, time-of-day, day-of-week, and data size.

An advantage of the invention within a business environment is that the method and system protect employee productivity by ensuring that Internet access is used primarily for business purposes. Another advantage is that the bandwidth availability is used more efficiently. Access may be dynamically controlled based upon factors such as the time of day and the day of the week. Another advantage is that internal security is enhanced by ensuring that access to internal computer resources is managed.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an exemplary topology of a network that utilizes access control management in accordance with the invention.

FIG. **2** is a block diagram of an exemplary network topology having more than one node that establishes access control in accordance with the invention.

FIG. **3** is a prior art schematic diagram of an Ethernet data packet.

FIG. **4** is a schematic view of the seven-layer ISO model and the source layers that are utilized by the invention.

FIG. **5** is a view of a graphical user interface (GUI) in accordance with one embodiment of rules configurations.

FIG. **6** is a block diagram of one embodiment of an access control device in accordance with the invention.

FIG. **7** is a process flow of steps for operating the device of FIG. **6**.

DETAILED DESCRIPTION

With reference to FIG. **1**, an exemplary network is shown as including a router **10** that provides access to the global communication network referred to as the Internet **14** for an organization that is protected from unwanted intruders by a firewall **16**. A number of conventional user workstations **18**, **20** and **22** are included as nodes of the network. A fourth workstation **24** may be identical to the other workstations, but is dedicated to providing access control management. Thus, the workstation **24** is an access control management console (ACMC). However, one of the other workstations may be used to implement the access rules in a manner that is consistent with the non-intrusive management system to be described below. The workstation **24** may be a conventional desktop computer having a plug-in access management module **26** to monitor traffic within the network.

Another node within the network is a proprietary proxy server **28** that is used in a conventional manner to enable selected services, such as web services. A web proxy server is designed to enable performance improvements by caching frequently accessed web pages. While such servers tend to add some access control potential by taking advantage of the fact that all HTTP conversions are being channeled through the service, the access control functionality is not a primary focus and only a subset of the protocols that are likely to be encountered via the Internet will be recognized by conventional web proxy servers. For example, the proxy server **28** may provide proxying capability for the HTTP protocol and perhaps browser-based FTP and Gopher, but the proxying capability is not likely to extend to other TCP/IP application protocols, such as telnet, news, e-mail and many proprietary multimedia protocols.

The network topology of FIG. **1** is shown as an exemplary configuration and is not meant to limit or constrain the description of the invention. The method and system to be described below can operate on a wide variety of network configurations. Moreover, while all workstations **18**–**24** can be presumed to be running the Microsoft Windows operating system and all servers **28** can be assumed to be running the Microsoft Windows NT Server operating system, the invention is not specific to any one operating system. Although the prime use of the method is anticipated as being applied to networks using the TCP/IP protocols, it can be readily adapted to function with any other set of networking protocols, such as Novell IPX/SPX or IBM NetBEUI.

It is also assumed that the network for which access management is to be provided includes a number of users, groups of users and workstation addresses. All of these items are assumed to have been pre-configured using known configuration methods provided by the supplier of the network operating system. Although implementation of the invention may be based on data such as usernames and group names from a network operating system or similar repository, there is no dependency on a specific network operating system or a specific mechanism to access such data. Employing usernames and group names that are consistent with other system operations takes advantage of any familiarity that may already exist with this information. Furthermore, in the absence of any such information, the invention may utilize other naming nomenclature, such as IP or Ethernet addresses.

Referring now to FIG. **2**, a first access control module **30** has been installed on the workstation **18** to enable the workstation to function as a passive access control station (PACS). A second instance of an access control module **32** is installed on the proxy server **28**, so that this node functions as a proxy access control station (PRACS). Moreover, a third instance of an access control module **34** is installed on the firewall **16** in order to form a gateway access control station (GACS). A key point in the system and method is that the individual workstations **20** and **22** that are accessed by users can be managed without installing any software components specifically on those workstations. Network traffic is monitored and access to internal and external resources is controlled and managed either at choke points (represented by the proxy server **28** and the firewall **16**) and/or non-intrusively at nodes which are not choke points (represented by the workstation **18**). The access control modules **30**, **32** and **34** can be installed, de-installed, and reinstalled on any of the nodes of the network at any time to suit potentially changing network topologies or changing access management policies.

The location and configuration of each of the access control modules **30**, **32** and **34** are selected by an installer based upon pragmatic factors in order to achieve a level of access control that is consistent with the access management policy. As previously noted, the first access control module **30** is not required, since the workstation **24** may serve the dual purpose of allowing a system operator to configure the rules base of access rules and non-intrusively monitoring traffic along the network. The second access control module **32** is optionally used in order to ensure that access is managed for all users who are accessing the WWW by configuring web browsers to operate via the proxy server **28**. The third access control module **34** is optionally installed at the firewall **16** in order to validate that both the firewall and the other access control modules have indeed been configured correctly and are performing their desired duties. Firewalls are sometimes difficult to configure, so organizations are increasingly adding second-line checks to their networks to ensure that absolute integrity is being maintained. However, the non-intrusive monitoring at the dedicated workstation **18** is capable of monitoring and controlling all access from all nodes on the network, regardless of TCP/IP protocol. This mechanism can be used to manage all network access that is not routed via the proxy server **28** with a high degree of probability that undesired access can indeed be blocked. Network traffic is non-intrusively monitored, but the system and method may be used to proactively block any requests for resources.

The non-intrusive monitoring of network traffic at the workstation **18** occurs by receiving and assembling data packets of node-to-node transmissions. Modern networks, including the Internet, are packet switching networks in which a transmission is separated into data packets which are separately transmitted to a destination node. At the destination node, the packets are assembled to form the original composite signal. FIG. **3** depicts an Ethernet data packet according to RFC base 894. Traffic along the network of FIGS. **1** and **2** may be in the form of transmissions of Ethernet packets. Each Ethernet packet **36** includes five segments. A first 6-byte segment **38** identifies the destination node address, while a second 6-byte segment **40** identifies the address of the source node. The third segment **42** is a 2-byte segment that identifies the protocol type. A data field **44** has a variable length, with a maximum of 1500 bytes. The data field **44** contains the user information. Finally, the fifth segment **46** is a checksum field that is used for error detection and correction purposes.

As is well known in the art, other standards for packetization are utilized. For example, each header that is used in a TCP transmission or a UDP (User Datagram Protocol) transmission includes a 16-bit destination port number. An Ethernet packet having a TCP/IP packet or UDP/IP packet embedded in its data field will include three designations: (1) the Ethernet addresses of the source and destination nodes; (2) the IP addresses of the source and destination nodes; and (3) the IP port number of the destination node. Other protocols are present and operational in TCP/IP networks and control operations such as routing and the translation of IP addresses to and from hostnames. A protocol referred to as ARP (Address Resolution Protocol) also maps IP addresses to Ethernet addresses.

By intercepting the Ethernet packet **36** of FIG. **3**, the destination address, the source address and the user data are available to the monitoring node. For the non-intrusive monitoring that occurs at the workstation **18** of FIG. **2**, the workstation may be placed in the promiscuous mode and there will be no impact on performance of the network. However, the packets that are specific to a particular node-to-node transmission can be collected and assembled merely by configuring the access control module **30** such that the workstation functions as a bare-bones TCP/IP protocol virtual machine. The workstation then has the capability of piecing together the fragments of a multi-packet signal. This enables access management control to base decisions upon information from various levels of the ISO model—from the lower layers to the uppermost Application Layer.

Communications protocols are a layered set, often referred to as a "stack." The International Standards Organization (ISO) has developed a model referred to as the ISO 7-layer model, which serves as a basic reference. Each layer represents a particular function. The function of a particular layer may be executed in hardware or software or a combination of hardware and software. At times, a single program performs the functions of more than one layer. FIG. **4** illustrates the seven layers of the ISO model. The lowermost layer, referred to as the Physical Layer **50**, is the hardware network connection, such as a physical wire. ISO Layer **2**, the Data Link Layer **52**, is responsible for providing reliable transmissions of data. Layer **2** may be a network interface card that links a computer to the network.

ISO Layer **3**, the Network Layer **54**, is the network software for routing packets throughout the network. ISO Layer **4**, the Transport Layer **56**, transports data from the network to the upper levels of the ISO model.

ISO Layer **5**, the Session Layer **58**, deals with establishing network sessions. Logical connections are established based upon a request of a user. ISO Layer **6**, the Presentation Layer **60**, deals with the presentation of data to an application which resides at ISO Layer **7**, the Application Layer **62**. Examples of the Application Layer include FTP, HTTP and SMTP. Layer **7** provides access to the Internet for a user.

FIG. **4** illustrates three inputs to a step **64** of storing data packets. The first input **66** represents the actual input of data packets, while the second and third inputs **68** and **70** are operational representations. Referring to FIGS. **2** and **4**, the workstation **18** that non-intrusively monitors network traffic receives inbound and outbound data packets through Layers **1** and **2**. As previously noted, the network interface card of Layer **2** is set to the promiscuous mode, so that the data packets of the network are received over the Physical Layer **50**. Optionally, the rules base of the access management module **26** may be utilized more than one time. In a first application of the rules base, the first packet of a resource request may be used to detect the source and destination nodes, allowing access determinations to be based on this low-level information. However, higher level decisions can be formed only after a connection has been established and the actual content has begun to flow over that connection. This is in contrast to conventional operations of firewalls, which typically only act as low-level packet filters (i.e., at ISO Layer **2**).

As indicated by the input **68**, the invention includes assembling the data packets to detect information at the Transport Layer **56** and the Network Layer **54** of the ISO model. Moreover, Layer **7** information is acquired by assembling the data packets, as represented by the input **70**. For example, in an e-mail environment, the Application Layer information that may be relevant to application of the rules base may include information within the "subject" line of an e-mail message. This information is acquired only upon accessing the data fields of the data packets of the e-mail message. At step **66**, the necessary information has been acquired for applying the rules base. As previously noted, the application may occur more than once for a single multi-packet transmission. The desirability of providing single or multiple rules applications may depend upon a number of factors.

Referring now to FIG. **5**, an embodiment of a graphical user interface (GUI) **68** is shown for use by a system operator to configure the rules base that determines the action of the access control modules **30**, **32** and **34** of FIG. **2**. The action of each access control module is determined by rules configured at the ACMC **24**, which includes the access management module **26**. The management module presents the GUI **68**, although this is not critical to the invention.

In the preferred embodiment, the rules base is comprised of a twin set of ordered rules. One of the sets of rules relates to access management requirements for outgoing access, while the second set relates to inbound connection attempts. Within each set, the rules are in a sequence that dictates the sequence in which the rules are considered. This sequencing ensures that rules are applied in a specific deterministic order, allowing the system operator to layer more specific rules ahead of more general rules. Thus, seemingly inconsistent rules can be established. For example, a rule may be configured to give User A access to a certain resource ahead of a rule banning everyone in the organization from accessing that resource. This has the effect of allowing access by User A and blocking access to that resource by all other users.

After a rules base has been configured by a system operator, the rules base is downloaded to the access control modules **30**, **32** and **34**. Thus, any subsequent changes in the rules base may be implemented at the various nodes in an efficient dynamic manner.

Regarding the configuration of the rules, various objects may be utilized to provide a more granular or less granular rule. Affected parties may be designated by usernames and group names (both typically from the network operating system), ad hoc groupings of users, and workstation addresses. Other objects include network services, source addresses (IP address, hostname or URL), destination addresses (IP address, hostname or URL) and time-slot specifiers (time of day, day of week, etc.). These objects are graphically dragged and dropped onto each rule, as required in order to dynamically and graphically build up the rule within the overall rules base. Against each rule, an action is configured to specify the resulting action that should be performed if a rule is matched at runtime. Potential actions include (1) disallowing the connection attempt, (2) allowing the connection attempt to be completed, (3) passing off the decision-making on whether the connection should be allowed or disallowed to a third-party component (which may, for example, consult a control list or perform other checks), (4) allowing the connection, but performing further analysis on the data stream in order to determine whether a connection should be broken at some future point (e.g., based upon the number of bytes that are transferred), and (5) performing further collection of the data stream and passing off the collection to a third-party component for further analysis (e.g., an anti-virus product).

Rules can be amended, deleted or reordered using the graphical user interface **68** of FIG. **5**. The rules base is stored in an internal format that is then made available to the various access control modules **30**, **32** and **34**, as described above.

The graphical user interface **68** is divided into two portions. The lower portion **70** is used to define network objects, such as usernames, groups, workstations and other such entities mentioned above. This information is built up by the system operator, but as much information as possible is gleaned from the network operating system. Typically, all usernames, group names and workstation addresses are established via reference to the network operating system. It is also possible to form ad hoc groupings for ease of use, such as groupings of users that are not configured or that are configured differently in the network operating system. Object-oriented technology simplifies the definition process by allowing operational parameters to be defined for object classes, rather than each individual network element. It is thus possible to perform access control at a detailed level of controlling individual user access and at a more general level of network groups of users or ad hoc groupings of users. This allows the operator to have flexibility in the access management task. It is thus possible to allow different access control criteria to different levels of employees and managers.

Other objects that are defined in the lower portion **70** of the GUI **68** are services, such as e-mail, file transfer, WWW and any of the other possible sets of services allowed in a TCP/IP network. Specific properties of a service include its name and its TCP/IP port number. Certain well-known services are pre-configured for the operator. For example, it is known that the telnet service should be pre-configured on port **23**. Any services may, however, be added or modified by the operator.

The upper portion **72** of the GUI **68** contains the rules. The total set of rules is referred to as the rules base. Rules are constructed graphically by the operator by dragging objects from the lower portion **70** and dropping them into specific rules of the upper portion **70**. Rule ordering is important and can be changed graphically by dragging a rule to a new position in the sequence. When rules are consulted at runtime, a top-down ordering is implemented. As previously noted, two sets of rules are maintained, one relating to outbound communications and the other relating to inbound communications.

In the preferred embodiment, storage logs are maintained for transaction data. The storage logs may be maintained for all of the transaction data or subsets of the data. The storage logs may then be used for further analysis by built-in or third-party components. However, this is not critical to the invention.

FIG. **6** is an exemplary arrangement of hardware and software for implementing the network access control system and method. A Passive Access Control Station, such as the workstation **18** of FIG. **2**, includes an input port **74** that is placed in a mode to receive all data packets destined for any node on the network. The data packets that are specific to a particular node-to-node transmission are combined at a packet assembler **76**. Detailed information from the assembled data packets is stored until sufficient information is acquired regarding the node-to-node transmission to apply the previously configured rules base **70**. The process of applying the rules base to the acquired information may occur in a single step or may be a multi-step process. For example, in FIG. **6** there is a state identifier **80** and a context identifier **82**. The state identifier is used to determine information regarding the lower layers of the ISO model, while the context identifier **82** acquires higher layer information, including Application Layer information. The rules base **78** may be consulted a first time when the state identifier **80** has acquired sufficient information, and then applied a second time when the context identifier **82** has acquired sufficient higher level information.

It is important to note that information which is stored includes both low level state information and contextual information that is discovered at points in the network stack other than Layers **1** and **2**. Full Application Layer awareness is achieved without the need to implement specific application proxies for each service. The two parts of the proxy process are linked in order to accommodate the possibility that proxy connections are being made, since the real source node and the final destination node must be identified to ensure that the correct rule is applied in managing network access.

If it is determined that a particular node-to-node transmission is unrestricted, the transmission is unaffected by the process. Optionally, data regarding the transmission may be stored within a log **84**. However, if the transmission is a restricted transmission, any one of a number of actions may be initiated by a connection controller **86**. When the connection from a source node to a destination node has not been completed, the connection controller may generate a signal that is output via the output port **88** to an appropriate node (e.g., a router) for preventing the connection. For situations in which the connection is established, the controller **86** may generate a signal that disables the connection. As a third alternative, the connection may be allowed, but further analysis of the data stream may be performed in order to ascertain whether the connection should be disabled at some future time (e.g., based upon the number of bytes that are transferred during the connection). The decision of whether to allow or disallow the connection may be passed to another node, such as a third-party component which consults a control list or performs other checks.

FIG. **7** details the steps of providing access control in accordance with the invention. In step **90**, network traffic is monitored non-intrusively, such as by the workstation **18** of FIG. **2**. Packets that are specific to a particular communication (i.e., node-to-node transmission) are identified in step **92** and assembled in step **94**. Decision step **96** determines whether sufficient information has been acquired to apply the rules of the rules base.

When sufficient information has been acquired to apply the rules base, the first rule is consulted to determine if the packet information set matches the rule. As previously noted, the rules base is organized into a first set of outbound-related rules and a second set of inbound-related rules. Moreover, the rules in a particular set are consulted in a top-down order. Thus, the rule that is applied in step **98** is the first rule in the appropriate set of rules. At step **100**, a decision is made as to whether the information set fits the rule applied in step **98**. If a rule fit is recognized, the appropriate rule action is applied at step **102**. The appropriate rule action may be designated within the rules base. If the rule is affirmatively stated (e.g., "allow all HTTP connections"), the action will allow the connection to remain unhindered. Other prescribed actions may include logging information to a database, sending an e-mail message, raising an alert in a pre-established manner, or diverting the data content of the connection to a third-party process which can determine whether the connection should be maintained by referencing other data, such as anti-virus rules or one or more control lists.

If in the decision step **100** it is determined that the first rule is not applicable, decision step **104** determines whether there is another applicable rule. If there are fifteen rules within the set of rules that are applicable to the communication under consideration, steps **98**, **100** and **104** will be repeated fifteen times or until the information set matches one of the rules.

Preferably, there is a default rule at the end of each set of rules in the rules base. Referring briefly to FIG. **5**, the GUI **68** shows six rules in its set of outgoing rules in the upper portion **72** of the GUI. The sixth and final rule to be applied is the default rule that disallows outgoing communications that are not specifically allowed within the set. Alternatively, the default rule may be to allow the communication.

After all of the appropriate rules have been applied, the optional decision step **106** is executed. The access rules of the rules base are pre-parsed to identify which rules can be applied at the basic connection time and which rules need to be held-over for application once the connection is completed and data is flowing. If, for a particular node-to-node transmission, it is determined that no rules need to be held-over, the default rule can be applied at connection time, assuming that there is no prior rule that provides an affirmative response at step **100**. However, if access rules need to be applied once data is flowing, the default rule is applied with the held-over rules. Thus, when there are access rules that relate to data flow, the connection is allowed to be completed, unless it is determined at step **100** that the connection is a restricted one. If it is determined at step **106** that rules have been held-over, the packets continue to be assembled at step **94** and the process repeats itself in order to apply the held-over rules. On the other hand, if there are no held-over rules, the process returns to the step **92** of identifying packets of a specific communication. However, the implementation, and even existence, of step **106** is not critical to the invention.

It is worth noting that various changes and modifications can be made to the above examples to achieve the same results, while remaining within the scope of the method and system. For example, access management control can be performed on a generic gateway machine, as opposed to a firewall, a proxy server or a passive workstation.

What is claimed:

1. A method of providing access control to resources of a network comprising steps of:

monitoring network traffic, including receiving data packets transmitted to and from nodes of said network such that receptions of said data packets are non-intrusive with respect to traffic flow of said network;

with respect to individual node-to-node transmissions within said network, assembling pluralities of said received data packets specific to said individual node-to-node transmissions, thereby forming an assembled multi-packet communication for each of said node-to-node transmission

based upon said assembled multi-packet communications, identifying source nodes and destination nodes and contextual information for said individual node-to-node transmissions; and

applying access rules to said assembled multi-packet communications in determinations of whether said individual node-to-node transmissions are restricted transmissions, including basing said determinations on said identifying said source and destination nodes and said contextual information.

2. The method of claim **1** wherein said steps of receiving and assembling said data packets are executed at a network element that is outside of direct paths from said source nodes to said destination nodes of said node-to-node transmissions.

3. The method of claim **2** wherein said steps of receiving and assembling said data packets are executed at a workstation that is dedicated to providing access control to said resources.

4. The method of claim **1** further comprising a step of determining whether to disallow said individual node-to-node transmissions based upon said step of applying said access rules.

5. The method of claim **1** further comprising a step of generating said access rules in a form of a rules base that includes a first set of rules specific to individual node-to-node transmissions having a destination node that is outside of said network and further includes a second set of rules specific to individual node-to-node transmissions having both of said source and destination nodes as network elements of said network, said step of applying said access rules including applying said first and second sets to user-generated information that is determined from said assembled multi-packet communications.

6. The method of claim **5** wherein said step of generating said access rules includes forming said first set of rules to be specific to communications via the global communications network referred to as the Internet and includes forming said second set of rules to be specific to communications that are on a same side of a firewall of said network.

7. The method of claim **6** wherein said step of assembling said received data packets is enabled for at least one of Transmission Control Protocol (TCP) services and User Datagram Protocol (UDP) services.

8. The method of claim **5** wherein said step of generating said access rules further comprises basing at least some of said access rules upon time, such that said determinations of whether said individual node-to-node transmissions are restricted transmissions are time-dependent determinations.

9. The method of claim **1** wherein said step of identifying said source and destination nodes and said contextual information includes collecting ISO Layer **7** data for use in said step of applying said access rules.

10. The method of claim **1** further comprising a step of executing first-line network intrusion detection at an entry point of said network, such that transmissions from nodes that are external to said network are subject to first-line network intrusion restriction rules, said first-line network intrusion detection being independent of said step of applying said access rules.

11. A method of providing access control to resources that are internal to and external of a network of nodes, including computing devices of users of said network, said method comprising steps of:

generating a rules base related to restricting access to said resources by said nodes of said network, including forming a first set of rules specific to access to external resources and a second set of rules specific to access to internal resources;

monitoring transmissions that include one of said computing devices;

acquiring information regarding each said transmission, including determining information relating to at least Layers **2**, **3** and **7** of the ISO model; and

applying said rules base to said acquired information to detect transmissions in which access to said resources is restricted by said rules base, including initiating a predetermined action in response to detecting that a specific transmission relates to an access that is restricted.

12. The method of claim **11** wherein said steps of monitoring said transmissions and acquiring said information are executed non-intrusively, such that transmissions for which accesses are restriction-free occur without impact on transmission traffic within said network.

13. The method of claim **12** wherein said steps of monitoring said transmissions and acquiring said information include receiving and assembling data packets at a node of said network, thereby forming a multi-packet communication for each said transmission, said node being outside of direct paths of said transmissions to and from said user computing devices.

14. The method of claim **11** wherein said step of acquiring information relating to Layer **7** includes assembling data packets received at said network via the Internet and includes assembling data packets that are exchanged between network elements of said network, said step of acquiring information further including determining user-generated contextual data relating to sources and destinations of said data packets.

15. A system for providing access control to resources of a network comprising:

a plurality of nodes, including computing devices;

means for non-intrusively intercepting data packets to and from said nodes such that said intercepting is substantially transparent to packet flow within said network;

means for identifying said data packets of discrete transmissions and assembling said data packets;

means for determining sources and destinations of said discrete transmissions and determining user-generated contextual information contained therein;

a rules base store having a plurality of rules relating to controlling access to said resources of said network; and

means for controlling said access based upon matching said rules to said sources, destinations and user-generated contextual information from said means for determining.

16. The system of claim **15** wherein said means for non-intrusively intercepting said data packets is positioned within said network and is operative to receive data packets transmitted between said nodes of said network.

17. The system of claim **16** wherein said means for non-intrusively intercepting is one of a workstation or a server dedicated to access control within said network, said workstation or server being on a same side of a firewall as said computing devices of said network.

18. The system of claim **16** wherein said rules base store includes a first set of rules specific to transmissions to destinations outside of said network and includes a second set of rules specific to transmissions having sources and destinations that are nodes of said network.

* * * * *

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L 29/06, 29/08 — A1

(11) International Publication Number: WO 98/34385

(43) International Publication Date: 6 August 1998 (06.08.98)

(21) International Application Number: PCT/US98/01797

(22) International Filing Date: 30 January 1998 (30.01.98)

(30) Priority Data:

60/036,661	30 January 1997 (30.01.97)	US
60/036,662	30 January 1997 (30.01.97)	US
08/818,769	14 March 1997 (14.03.97)	US

(71) Applicant: VXTREME, INC. [US/US]; 675 Almanor Avenue, Sunnyvale, CA 94086 (US).

(72) Inventors: VELLANKI, Srinivas, Prasad; 400 Glenmoor Circle, Milpitas, CA 95035 (US). CANNON, Anthony, William; Apartment 334, 1600 Villa Street, Mountain View, CA 94041 (US). RAVI, Hemanth, Srinivas; 444 Glenmoor Circle, Milpitas, CA 95035 (US). KLEMETS, Anders, Edgar; 655 S. Fair Oaks Avenue, #K-313, Sunnyvale, CA 94086 (US).

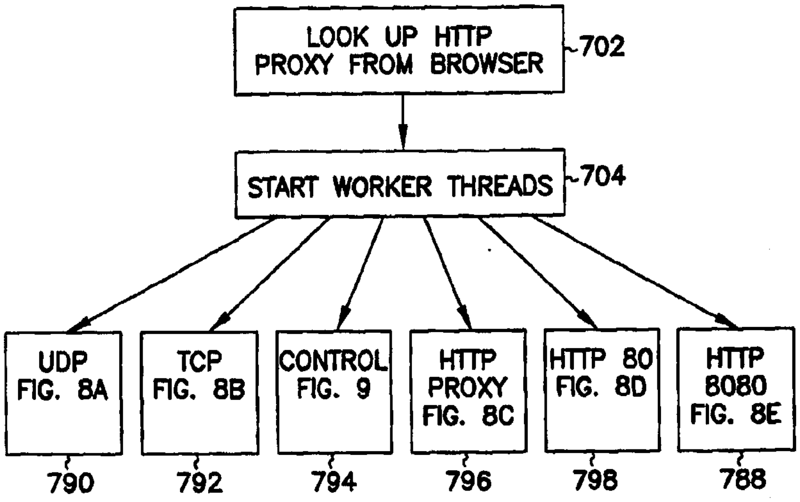
(74) Agent: VIKSNINS, Ann, S.; Schwegman, Lundberg, Woessner & Kluth, P.O. Box 2938, Minneapolis, MN 55402 (US).

(81) Designated States: JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published

With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: AUTOMATIC DETECTION OF PROTOCOLS IN A NETWORK



(57) Abstract

A method in a computer network for automatically detecting a most advantageous protocol for communication by a client computer, said client computer being configured to be coupled to a server computer via a computer network. The method includes initiating a plurality of protocol threads for sending from the client computer to the server computer, a plurality of data requests. Each of the data requests employs a different protocol and a different connection. The data requests are configured to solicit, responsive to the data requests, a set of responses from the server computer. Each of the responses employs a protocol associated with a respective one of the data requests. The method further includes receiving at the client computer at least a subset of the responses. The method also includes initiating a control thread at the client computer. The control thread monitors the subset of the responses as each response is received from the server computer to select the most advantageous protocol from protocols associated with the subset of the responses, wherein the most advantageous protocol is determined based on a predefined protocol priority.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania
AM	Armenia
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CU	Cuba
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GH	Ghana
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IL	Israel
IS	Iceland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakstan
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
MK	The former Yugoslav Republic of Macedonia
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TM	Turkmenistan
TR	Turkey
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam
YU	Yugoslavia
ZW	Zimbabwe

AUTOMATIC DETECTION OF PROTOCOLS IN A NETWORK

BACKGROUND OF THE INVENTION

The present invention relates to data communication in a computer network. More particularly, the present invention relates to improved methods and apparatus for permitting a client computer in a client-server architecture computer network to automatically detect the most advantageous protocol, among the protocols available, for use in communicating with the server irrespective whether there exist firewalls or proxies in the network.

Client-server architectures are well known to those skilled in the computer art. For example, in a typical computer network, one or more client computers may be coupled to any number of server computers. Client computers typically refer to terminals or personal computers through which end users interact with the network. Server computers typically represent nodes in the computer network where data, application programs, and the like reside. Server computers may also represent nodes in the network for forwarding data, programs, and the like from other servers to the requesting client computers.

To facilitate discussion, Fig. 1 illustrates a computer network 100, representing for example a subset of an international computer network popularly known as the Internet. As is well known, the Internet represents a well-known international computer network that links, among others, various military, governmental, educational, nonprofit, industrial and financial institutions, commercial enterprises, and individuals. There are shown in Fig. 1 a server 102, a server 104, and a client computer 106. Server computer 104 is separated from client computer 106 by a firewall 108, which may be implemented in either software or hardware, and may reside on a computer and/or circuit between client computer 106 and server computer 104.

Firewall 108 may be specified, as is well known to those skilled in the art, to prevent certain types of data and/or protocols from traversing through it. The specific data and/or protocols prohibited or permitted to traverse firewall

108 depend on the firewall parameters, which are typically set by a system administrator responsible for the maintenance and security of client computer 106 and/or other computers connected to it, e.g., other computers in a local area network. By way of example, firewall 108 may be set up to prevent TCP, UDP, or HTTP (Transmission Control Protocol, User Datagram Protocol, and Hypertext Transfer Protocol, respectively) data and/or other protocols from being transmitted between client computer 106 and server 104. The firewalls could be configured to allow specific TCP or UDP sessions, for example outgoing TCP connection to certain ports, UDP sessions to certain ports, and the like.

Without a firewall, any type of data and/or protocol may be communicated between a client computer and a server computer if appropriate software and/or hardware are employed. For example, server 102 resides on the same side of firewall 108 as client computer 106, i.e., firewall 108 is not disposed in between the communication path between server 102 and client computer 106. Accordingly, few if any, of the protocols that client computer 106 may employ to communicate with server 102 may be blocked.

As is well known to those skilled in the art, some computer networks may be provided with proxies, i.e., software codes or hardware circuities that facilitate the indirect communication between a client computer and a server around a firewall. With reference to Fig. 1, for example, client computer 106 may communicate with server 104 through proxy 120. Through proxy 120, HTTP data, which may otherwise be blocked by firewall 108 for the purpose of this example, may be transmitted between client computer 106 and server computer 104.

In some computer networks, one or more protocols may be available for communication between the client computer and the server computer. For certain applications, one of these protocols, however, is often more advantageous, i.e., suitable, than others. By way of example, in applications involving real-time data rendering (such as rendering audio, video, and/or annotation data as they are streamed from a server), it is highly preferable that the client computer executing that application selects a protocol that permits the greatest degree of control over the transmission of data packets and/or enables data transmission to occur at the highest possible rate. This is because these applications are fairly demanding in terms of their bit rate and connection reliability requirements. Accordingly, the quality of the data rendered, e.g., the video and/or audio clips played, often depends on whether the user has successfully configured the client computer to receive data from the server computer using the most advantageous protocol available.

In the prior art, the selection of the most advantageous protocol for communication between client computer 106 and server computer 104 typically requires a high degree of technical sophistication on the part of the user of client computer 106. By way of example, it is typically necessary in the prior art for the user of client computer 106 to understand the topology of computer network 100, the protocols available for use with the network, and/or the protocols that can traverse firewall 108 before that user can be expected to configure his client computer 106 for communication.

This level of technical sophistication is, however, likely to be beyond that typically possessed by an average user of client computer 106. Accordingly, users in the prior art often find it difficult to configure their client computers even for simple communication tasks with the network. The difficulties may be encountered for example during thc initial setup or whenever there are changes in the topology of computer network 100 and/or in the technology employed to transmit data between client computer 106 and server 104. Typically, expert and expensive assistance is required, if such assistance is available at all in the geographic area of the user.

Furthermore, even if the user can configure client computer 106 to communicate with server 104 through firewall 108 and/or proxy 120, there is no assurance that the user of client computer 106 has properly selected, among the protocols available, the most advantageous protocol communication (e.g., in terms of data transmission rate, transmission control, and the like). As mentioned earlier, the ability to employ the most advantageous protocol for communication, while desirable for most networking applications, is particularly critical in applications such as real-time data rendering (e.g., rendering of audio, video, and/or annotation data as they are received from the server). If a less than optimal protocol is chosen for communication, the quality of the rendered data, e.g., the video clips and/or audio clips, may suffer.

In view of the foregoing, there are desired improved techniques for permitting a client computer in a client-server network to efficiently, automatically, and appropriately select the most advantageous protocol to communicate with a server computer.

SUMMARY OF THE INVENTION

The invention relates, in one embodiment, to a method in a computer network for automatically detecting a most advantageous protocol for communication by a client computer. The client computer is configured to be coupled to a server computer via a computer network. The method includes sending from the client computer to the server computer, a plurality of data requests. Each of the data requests employs a different protocol and a different connection. The data requests are configured to solicit, responsive to the data requests, a set of responses from the server computer. Each of the responses employs a protocol associated with a respective one of the data requests.

The method further includes receiving at the client computer at least a subset of the responses. The method also includes monitoring the subset of the responses as each response is received from the server computer to select the most advantageous protocol from protocols associated with the subset of the responses, wherein the most advantageous protocol is determined based on a predefined protocol priority.

In another embodiment, the invention relates to a method in a computer network for automatically detecting a most advantageous protocol for communication by a client computer. The client computer is configured to be coupled to a server computer via a computer network. The method includes sending from the client computer to the server computer, a plurality of data

requests. Each of the data requests employs a different protocol and a different connection.

The method further includes receiving at least a subset of the data requests at the server computer. The method additionally includes sending a set of responses form the server computer to the client computer. The set of responses is responsive to the subset of the data requests. Each of the responses employs a protocol associated with a respective one of the subset of the data requests. The method also includes receiving at the client computer at least a subset of the responses. There is further included selecting, for the communication between the client computer and the server computer, the most advantageous protocol from protocols associated with the subset of the responses, wherein the most advantageous protocol is determined based on a predefined protocol priority.

In yet another embodiment, the invention relates to a computer-readable medium containing computer-readable instructions for automatically detecting a most advantageous protocol for communication by a client computer. The client computer is configured to be coupled to a server computer via a computer network. The computer-readable instructions comprise computer-readable instructions for sending in a substantially parallel manner, from the client computer to the server computer, a plurality of data requests. Each of the data requests employs a different protocol and a different connection. The data requests are configured to solicit, responsive to the data requests, a set of responses from the server computer. Each of the responses employs a protocol associated with a respective one of the data requests.

The computer-readable medium further includes computer-readable instructions for receiving at the client computer at least a subset of the responses. There is further included computer-readable instructions for monitoring the subset of the responses as each response is received from the server computer to select the most advantageous protocol from protocols associated with the subset of the responses, wherein the most advantageous protocol is determined based on a predefined protocol priority.

These and other features of the present invention will be described in more detail below in the detailed description of the invention and in conjunction with the following figures.

BRIEF DESCRIPTION OF THE DRAWINGS

To facilitate discussion, Fig. 1 illustrates a computer network, representing for example a portion of an international computer network popularly known as the Internet.

Fig. 2 is a block diagram of an exemplar computer system for carrying out the autodetect technique according to one embodiment of the invention.

Fig. 3 illustrates, in accordance with one embodiment, the control and data connections between a client application and a server computer when no firewall is provided in the network.

Fig. 4 illustrates another network arrangement wherein control and data connections are established through a firewall.

Figs. 5A-B illustrate another network arrangement wherein media control commands and media data may be communicated between a client computer and server computer using the HTTP protocol.

Figs. 5C-D illustrate another network arrangement wherein multiple HTTP control and data connections are multiplexed through a single HTTP port.

Fig. 6 illustrates another network arrangement wherein control and data connections are transmitted between the client application and the server computer via a proxy.

Fig. 7 depicts, in accordance with one embodiment of the present invention, a simplified flowchart illustrating the steps of the inventive autodetect technique.

Fig. 8A depicts, in accordance with one aspect of the present invention, the steps involved in executing the UDP protocol thread of Fig. 7.

Fig. 8B depicts, in accordance with one aspect of the present invention, the steps involved in executing the TCP protocol thread of Fig. 7.

Fig. 8C depicts, in accordance with one aspect of the present invention, the steps involved in executing the HTTP protocol thread of Fig. 7.

Fig. 8D depicts, in accordance with one aspect of the present invention, the steps involved in executing the HTTP 80 protocol thread of Fig. 7.

Fig. 8E depicts, in accordance with one aspect of the present invention, the steps involved in executing the HTTP 8080 protocol thread of Fig. 7.

Fig. 9 illustrates, in accordance with one embodiment of the present invention, the steps involved in executing the control thread of Fig. 7.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention will now be described in detail with reference to a few preferred embodiments thereof as illustrated in the accompanying drawings. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without some or all of these specific details. In other instances, well-known process steps have not been described in detail in order to not unnecessarily obscure the present invention.

In accordance with one aspect of the present invention, the client computer in a heterogeneous client-server computer network (e.g., client computer 106 in Fig. 1) is provided with an autodetect mechanism. When executed, the autodetect mechanism advantageously permits client computer 106 to select, in an efficient and automatic manner, the most advantageous protocol for communication between the client computer and its server. Once the most advantageous protocol is selected, parameters pertaining to the selected protocol are saved to enable the client computer, in future sessions, to employ the same selected protocol for communication.

In accordance with one particular advantageous embodiment, the inventive autodetect mechanism simultaneously employs multiple threads, through multiple connections, to initiate communication with the server computer, e.g., server 104. Each thread preferably employs a different protocol and requests the server computer to respond to the client computer using the protocol associated with that thread. For example, client computer 106 may, employing the autodetect mechanism, initiate five different threads, using respectively the TCP, UDP, one of HTTP and HTTP proxy, HTTP through port (multiplex) 80, and HTTP through port (multiplex) 8080 protocols to request server 104 to respond.

Upon receiving a request, server 104 responds with data using the same protocol as that associated with the thread on which the request arrives. If one or more protocols is blocked and fails to reach server 104 (e.g., by a firewall), no response employing the blocked protocol would of course be transmitted from server 104 to client computer 106. Further, some of the protocols transmitted from server 104 to client computer 106 may be blocked as well. Accordingly, client computer may receive only a subset of the responses sent from server 104.

In one embodiment, client computer 106 monitors the set of received responses. If the predefined "best" protocol is received, that protocol is then selected for communication by client computer 106. The predefined "best" protocol may be defined in advance by the user and/or the application program. If the predefined "best" protocol is, however, blocked (as the request is transmitted from the client computer or as the response is transmitted from the server, for example) the most advantageous protocol may simply be selected from the set of protocols received back by the client computer. In one embodiment, the selection may be made among the set of protocols received back by the client computer within a predefined time period after the requests are sent out in parallel.

The selection of the most advantageous protocol for communication among the protocols received by client computer 106 may be performed in accordance with some predefined priority. For example, in the real-time data rendering application, the UDP protocol may be preferred over TCP protocol, which may in turn be preferred over the HTTP protocol. This is because the UDP protocol typically can handle a greater data transmission rate and may allow client computer 106 to exercise a greater degree of control over the transmission of data packets.

HTTP data, while popular nowadays for use in transmitting web pages, typically involves a higher number of overhead bits, making it less efficient relative to the UDP protocol for transmitting real-time data. As is known, the HTTP protocol is typically built on top of TCP. The underlaying TCP protocol typically handles the transmission and retransmission requests of individual data packets automatically. Accordingly, the HTTP protocol tends to reduce the degree of control client computer 106 has over the transmission of the data packets between server 104 and client computer 106. Of course other priority schemes may exist for different applications, or even for different real-time data rendering applications.

In one embodiment, as client computer 106 is installed and initiated for communication with server 104 for the first time, the autodetect mechanism is invoked to allow client computer 106 to send transmission requests in parallel (e.g., using different protocols over different connections) in the manner discussed earlier. After server 104 responds with data via multiple connections/protocols and the most advantageous protocol has been selected by client computer 106 for communication (in accordance with some predefined priority), the parameters associated with the selected protocol are then saved for future communication.

Once the most advantageous protocol is selected, the autodetect mechanism may be disabled, and future communication between client computer 106 and server 104 may proceed using the selected most advantageous protocol without further invocation of the autodetect mechanism. If the topology of computer network 100 changes and communication using the previously selected "most advantageous" protocol is no longer appropriate, the autodetect mechanism may be executed again to allow client computer 106 to ascertain a new "most advantageous" protocol for communication with server 104. In one embodiment, the user of client computer 106 may, if desired, initiate the autodetect mechanism at any time in order to enable client computer 106 to update the "most advantageous" protocol for communication with server 104 (e.g., when the user of client computer 106 has reasons to suspect that the previously selected "most advantageous" protocol is no longer the most optimal protocol for communication).

The inventive autodetect mechanism may be implemented either in software or hardware, e.g., via an IC chip. If implemented in software, it may be carried out by any number of computers capable of functioning as a client computer in a computer network. Fig. 2 is a block diagram of an exemplar computer system 200 for carrying out the autodetect technique according to one embodiment of the invention. Computer system 200, or an analogous one, may be employed to implement either a client or a server of a computer network. The computer system 200 includes a digital computer 202, a display screen (or monitor) 204, a printer 206, a floppy disk drive 208, a hard disk drive 210, a network interface 212, and a keyboard 214. The digital computer 202 includes a microprocessor 216, a memory bus 218, random access memory (RAM) 220, read only memory (ROM) 222, a peripheral bus 224, and a keyboard controller 226. The digital computer 200 can be a personal computer (such as an Apple computer, e.g., an Apple Macintosh, an IBM personal computer, or one of the compatibles thereof), a workstation computer (such as a Sun Microsystems or Hewlett-Packard workstation), or some other type of computer.

The microprocessor 216 is a general purpose digital processor which controls the operation of the computer system 200. The microprocessor 216 can be a single-chip processor or can be implemented with multiple components. Using instructions retrieved from memory, the microprocessor 216 controls the reception and manipulation of input data and the output and display of data on output devices.

The memory bus 218 is used by the microprocessor 216 to access the RAM 220 and the ROM 222. The RAM 220 is used by the microprocessor 216 as a general storage area and as scratch-pad memory, and can also be used to store input data and processed data. The ROM 222 can be used to store instructions or program code followed by the microprocessor 216 as well as other data.

The peripheral bus 224 is used to access the input, output, and storage devices used by the digital computer 202. In the described embodiment, these devices include the display screen 204, the printer device 206, the floppy disk drive 208, the hard disk drive 210, and the network interface 212, which is employed to connect computer 200 to the network. The keyboard controller 226 is used to receive input from keyboard 214 and send decoded symbols for each pressed key to microprocessor 216 over bus 228.

The display screen 204 is an output device that displays images of data provided by the microprocessor 216 via the peripheral bus 224 or provided by other components in the computer system 200. The printer device 206 when operating as a printer provides an image on a sheet of paper or a similar surface. Other output devices such as a plotter, typesetter, etc. can be used in place of, or in addition to, the printer device 206.

The floppy disk drive 208 and the hard disk drive 210 can be used to store various types of data. The floppy disk drive 208 facilitates transporting such data to other computer systems, and hard disk drive 210 permits fast access to large amounts of stored data.

The microprocessor 216 together with an operating system operate to execute computer code and produce and use data. The computer code and data may reside on the RAM 220, the ROM 222, the hard disk drive 220, or even on another computer on the network. The computer code and data could also reside on a removable program medium and loaded or installed onto the computer system 200 when needed. Removable program mediums include, for example, CD-ROM, PC-CARD, floppy disk and magnetic tape.

The network interface circuit 212 is used to send and receive data over a network connected to other computer systems. An interface card or similar device and appropriate software implemented by the microprocessor 216 can be used to connect the computer system 200 to an existing network and transfer data according to standard protocols.

The keyboard 214 is used by a user to input commands and other instructions to the computer system 200. Other types of user input devices can also be used in conjunction with the present invention. For example, pointing devices such as a computer mouse, a track ball, a stylus, or a tablet can be used to manipulate a pointer on a screen of a general-purpose computer.

The invention can also be embodied as computer-readable code on a computer-readable medium. The computer-readable medium is any data storage device that can store data which can thereafter be read by a computer system. Examples of the computer-readable medium include read-only memory, random-access memory, CD-ROMs, magnetic tape, optical data storage devices. The computer-readable code can also be distributed over a network coupled computer system so that the computer-readable code is stored and executed in a distributed fashion.

Figs. 3-6 below illustrate, to facilitate discussion, some possible arrangements for the transmission and receipt of data in a computer network. The arrangements differ depending on which protocol is employed and the configuration of the network itself. Fig. 3 illustrates, in accordance with one embodiment, the control and data connections between a client application 300 and server 302 when no firewall is provided in the network.

Client application 300 may represent, for example, the executable codes for executing a real-time data rendering program such as the Web Theater Client 2.0, available from VXtreme, Inc. of Sunnyvale, California. In the example of Fig. 3, client application 300 includes the inventive autodetect mechanism and may represent a plug-in software module that may be installed onto a browser 306. Browser 306 may represent, for example, the application program which the user of the client computer employs to navigate the network. By way of example, browser 306 may represent one of the popular Internet browser programs, such as Netscape™ by Netscape Communications Inc. of Mountain View, California or Microsoft Explorer by Microsoft Corporation of Redmond, Washington.

When the autodetect mechanism of client application 300 is executed in browser 306 (e.g., during the set up of client application 300), client application 300 sends a control request over control connection 308 to server 302. Although multiple control requests are typically sent in parallel over multiple control connections using different protocols as discussed earlier, only one control request is depicted in Fig. 3 to facilitate ease of illustration.

The protocol employed to send the control request over control connection 308 may represent, for example, TCP, or HTTP. If UDP protocol is requested from the server, the request from the client may be sent via the control connection using for example the TCP protocol. Initially, each control request from client application 300 may include, for example, the server name that identifies server 302, the port through which control connection may be established, and the name of the video stream requested by client application 300. Server 302 then responds with data via data connection 310.

In Fig. 3, it is assumed that no proxies and/or firewalls exist. Accordingly, server 302 responds using the same protocol as that employed in the request. If the request employs TCP, however, server 302 may attempt to respond using either UDP or TCP data connections (depending on the specifics of the request). The response is sent to client application via data connection 310. If the protocol received by the client application is subsequently selected to be the "most advantageous" protocol, subsequent communication between client application 300 and server 302 may take place via control connection 308 and data connection 310. Subsequent control requests sent by client application 300 via control connection 308 may include, for example, stop, play, fast forward, rewind, pause, unpause, and the like. These control requests may be utilized by server 302 to control the delivery of the data stream from server 302 to client application 300 via data connection 310.

It should be noted that although only one control connection and one data connection is shown in Fig. 3 to simplify illustration, multiple control and data connections utilizing the same protocol may exist during a data rendering session. Multiple control and data connections may be required to handle the multiple data streams (e.g., audio, video, annotation) that may be needed in a particular data rendering session. If desired, multiple client applications 300 may be installed within browser 306, e.g., to simultaneously render multiple video clips, each with its own sound and annotations.

Fig. 4 illustrates another network arrangement wherein control and data connections are established through a firewall. As mentioned earlier, a firewall may have policies that restrict or prohibit the traversal of certain types of data and/or protocols. In Fig. 4, a firewall 400 is disposed between client application 300 and server 402. Upon execution, client application 300 sends control request using a given protocol via firewall 400 to server 402. Server 402 then responds with data via data connection 410, again via firewall 400.

If the data and/or protocol can be received by the client computer through firewall 400, client application 300 may then receive data from server 402 (through data connection 408) in the same protocol used in the request. As before, if the request employs the TCP protocol, the server may respond with data connections for either TCP or UDP protocol (depending on the specifics of the request). Protocols that may traverse a firewall may include one or more of the following: UDP, TCP, and HTTP.

In accordance with one aspect of the present invention, the HTTP protocol may be employed to send/receive media data (video, audio, annotation, or the like) between the client and the server. Fig. 5A is a prior art drawing illustrating how a client browser may communicate with a web server using a port designated for communication. In Fig. 5, there is shown a web server 550, representing the software module for serving web pages to a browser application 552. Web server 550 may be any of the commercially available web servers that are available from, for example, Netscape Communications Inc. of Mountain View, California or Microsoft Corporation of Redmond, Washington. Browser application 552 represents for example the Netscape browser from the aforementioned Netscape Communications, Inc., or similarly suitable browser applications.

Through browser application 552, the user may, for example, obtain web pages pertaining to a particular entity by sending an HTTP request (e.g., GET) containing the URL (uniform resource locator) that identifies the web page file. The request sent via control connection 553 may arrive at web server 550 through the HTTP port 554. HTTP port 554 may represent any port through which HTTP communication is enabled. HTTP port 554 may also represent the default port for communicating web pages with client browsers. The HTTP default port may represent, for example, either port 80 or port 8080 on web server 550. As is known, one or both of these ports on web server 550 may be available for web page communication even if there are firewalls disposed between the web server 550 and client browser application 552, which otherwise block all HTTP traffic in other ports. Using the furnished URL, web server 550 may then obtain the desired web page(s) for sending to client browser application 552 via data connection 556.

The invention, in one embodiment, involves employing the HTTP protocol to communicate media commands from a browser application or browser plug-in to the server. Media commands are, for example, PLAY, STOP, REWIND, FAST FORWARD, and PAUSE. The server computer may represent, for example, a web server. The server computer may also represent a video server for streaming video to the client computer. Through the use of the HTTP protocol the client computer may successfully send media control requests and receive media data through any HTTP port. If the default HTTP port, e.g., port 80 or 8080, is specified, the client may successfully send media control requests and receive media data even if there exists a firewall or an HTTP Proxy disposed in between the server computer and the client computer, which otherwise blocks all other traffic that does not use the HTTP protocol. For example, these firewalls or HTTP Proxies do not allow regular TCP or UDP packets to go through.

As is well known to those skilled, the HTTP protocol, as specified by for example the Internet Request For Comments RFC 1945 (T. Berners-Lee et al.), typically defines only three types of requests to be sent from the client computer to the server, namely GET, POST, and HEAD. The POST command, for instance, is specified in RFC 1945 to be composed of a Request-Line, one or more Headers and Entity-Body. To send media commands like PLAY, REWIND, etc., the invention in one embodiment sends the media command as part of the Entity-Body of the HTTP POST command. The media command can be in any format or protocol, and can be, for instance, in the same format as that employed when firewalls are not a concern and plain TCP protocol can be used. This format can be, for example, RTSP (Real Time Streaming Protocol).

When a server gets an HTTP request, it answers the client with an HTTP Response. Responses are typically composed of a Status-Line, one or more headers, and an Entity-Body. In one embodiment of this invention, the response to the media commands is sent as the Entity-Body of the response to the original HTTP request that carried the media command.

Fig. 5B illustrates this use of HTTP for sending arbitrary media commands. In Fig. 5B, the plug-in application 560 within client browser application 562 may attempt to receive media data (e.g., video, audio, annotation, or the like) by first sending an HTTP request to server 564 via control connection 565. For example, a REWIND command could be sent from the client 560 to the server 564 as an HTTP packet 570 of the form: "POST/HTTP/1.0<Entity-Body containing rewind command in any suitable media protocol>". The server can answer to this request with an HTTP response 572 of the form: "HTTP/1.0 200ok<Entity-Body containing rewind response in any suitable media protocol>".

The HTTP protocol can be also used to send media data across firewalls. The client can send a GET request to the video server, and the video server can then send the video data as the Entity-Body of the HTTP response to this GET request.

Some firewalls may be restrictive with respect to HTTP data and may permit HTTP packets to traverse only on a certain port, e.g., port 80 and/or port 8080. Fig. 5C illustrates one such situation. In this case, the control and data communications for the various data stream, e.g., audio, video, and/or annotation associated with different rendering sessions (and different clients) may be multiplexed using conventional multiplexer code and/or circuit 506 at client application 300 prior to being sent via port 502 (which may represent, for example, HTTP port 80 or HTTP port 8080). The inventive combined use of the HTTP protocol and of the multiplexer for transmitting media control and data is referred to as the HTTP multiplex protocol, and can be used to send this data across firewalls that only allow HTTP traffic on specific ports, e.g., port 80 or 8080.

At server 402, representing, for example, server 104 of Fig. 1, conventional demultiplexer code and/or circuit 508 may be employed to decode the received data packets to identify which stream the control request is associated with. Likewise, data sent from server 402 to client application 300 may be multiplexed in advance at server 402 using for example conventional multiplexer code and/or circuit 510. The multiplexed data is then sent via port 502. At client application 300, the multiplexed data may be decoded via conventional demultiplexer code and/or circuit 512 to identify which stream the received data packets is associated with (audio, video, or annotation).

Multiplexing and demultiplexing at the client and/or server may be facilitated for example by the use of the Request-URL part of the Request-Line of HTTP requests. As mentioned above, the structure of HTTP requests is described in RFC 1945. The Request-URL may, for example, identify the stream associated with the data and/or control request being transmitted. In one embodiment, the additional information in the Request-URL in the HTTP header may be as small as one or a few bits added to the HTTP request sent from client application 300 to server 402.

To further facilitate discussion of the inventive HTTP multiplexing technique, reference may now be made to Fig. 5D. In Fig. 5D, the plug-in application 660 within client plug-in application 660 may attempt to receive media data (e.g., video, audio, annotation, or the like) by first sending a control request 670 to server 664 via control connection 665. The control request is an HTTP request, which arrives at the HTTP default port 654 on server 664. As mentioned earlier, the default HTTP port may be either port 80 or port 8080 in one embodiment.

In one example, the control request 670 from client plug-in 660 takes the form of a command to "POST/12469 HTTP/1.0<Entity-Body>" which indicates to the server (through the designation 12469 as the Request-URL) that this is a control connection. The Entity-Body contains, as described above, binary data that informs the video server that the client plug-in 660 wants to display a certain video or audio clip. Software codes within server 664 may be employed to assign a unique ID to this particular request from this particular client.

For discussion sake, assume that server 664 associates unique ID 35,122 with a video data connection between itself and client plug-in application 660, and unique ID 29,999 with an audio data connection between itself and client plug-in application. The unique ID is then communicated as message 672 from server 664 to client plug-in application 660, again through the aforementioned HTTP default port using data connection 667. The Entity-Body of message 672 contains, among other things and as depicted in detail 673, the audio and/or video session ID. Note that the unique ID is unique to each data connection (e.g., each of the audio, video, and annotation connections) of each client plug-in application (since there may be multiple client plug-in applications attempting to communicate through the same port).

Once the connection is established, the same unique ID number is employed by the client to issue HTTP control requests to server 664. By way of example, client plug-in application 660 may issue a command "GET/35,122 HTTP/1.0" or "POST/35,122 HTTP/1.0<Entity-Body containing binary data with the REWIND media command>" to request a video file or to rewind on the video file. Although the rewind command is used in Figs. 5A-5D to facilitate ease of discussion, other media commands, e.g., fast forward, pause, real-time play, live-play, or the like, may of course be sent in the Entity-Body. Note that the unique ID is employed in place of or in addition to the Request-URL to qualify the Request-URL.

Once the command is received by server 664, the unique ID number (e.g., 35,122) may be employed by the server to demultiplex the command to associate the command with a particular client and data file. This unique ID number can also attach to the HTTP header of HTTP responses set from server 664 to client plug-in application 660, through the same HTTP default port 654 on server 664, to permit client plug-in application 660 to ascertain whether an HTTP data packet is associated with a given data stream.

Advantageously, the invention permits media control commands and media data to be communicated between the client computer and the server computer via the default HTTP port, e.g., port 80 or 8080 in one embodiment. even if HTTP packets are otherwise blocked by a firewall disposed between the client computer and the server computer. The association of each control connection and data connection to each client with a unique ID advantageously permits multiple control an data connections (from one or more clients) to be established through the same default HTTP port on the server, advantageously bypassing the firewall. Since both the server and the client have the demultiplexer code and/or circuit that resolve a particular unique ID into a particular data stream, multiplexed data communication is advantageously facilitated thereby.

In some networks, it may not be possible to traverse the firewall due to stringent firewall policies. As mentioned earlier, it may be possible in these situations to allow the client application to communicate with a server using a proxy. Fig. 6 illustrates this situation wherein client application 300 employs proxy 602 to communicate with server 402. The use of proxy 602 may be necessary since client application 300 may employ a protocol which is strictly prohibited by firewall 604. The identity of proxy 602 may be found in browser program 306, e.g., Netscape as it employs the proxy to download its web pages, or may be configured by the user himself. Typical protocols that may employ a proxy for communication, e.g., proxy 602, includes HTTP and UDP.

In accordance with one embodiment of the present invention, the multiple protocols that may be employed for communication between a server computer and a client computer are tried in parallel during autodetect. In other words, the connections depicted in Figs. 3, 4, 5C and 6 may be attempted simultaneously and in parallel over different control connections by the client computer. Via these control connections, the server is requested to respond with various protocols.

If the predefined "best" protocol (predetermined in accordance with some predefined protocol priority) is received by the client application from the server, autodetect may, in one embodiment, end immediately and the "best" protocol is selected for immediate communication. In one real-time data rendering application, UDP is considered the "best" protocol, and the receipt of UDP data by the client may trigger the termination of the autodetect.

If the "best" protocol has not been received after a predefined time period, the most advantageous protocol (in terms of for example data transfer rate and/or transmission control) is selected among the set of protocols received by the client. The selected protocol may then be employed for communication between the client and the server.

Fig. 7 depicts, in accordance with one embodiment of the present invention, a simplified flowchart illustrating the steps of the inventive autodetect technique. In Fig. 7, the client application starts (in step 702) by looking up the HTTP proxy, if there is any, from the browser. As stated earlier, the client computer may have received a web page from the browser, which implies that the HTTP protocol may have been employed by the browser program for communication. If an HTTP proxy is required, the name and location of the HTTP proxy is likely known to the browser, and this knowledge may be subsequently employed by the client to at least enable communication with the server using the HTTP proxy protocol, i.e., if a more advantageous protocol cannot be ascertained after autodetect.

In step 704, the client begins the autodetect sequence by starting in parallel the control thread 794, along with five protocol threads 790, 792, 796, 798, and 788. As the term is used herein, parallel refers to both the situation wherein the multiple protocol threads are sent parallely starting at substantially the same time (having substantially similar starting time), and the situation wherein the multiple protocol threads simultaneously execute (executing at the same time), irrespective when each protocol thread is initiated. In the latter case, the multiple threads may have, for example, staggered start time and the initiation of one thread may not depend on the termination of another thread.

Control thread 794 represents the thread for selecting the most advantageous protocol for communication. The other protocol threads 790, 792, 796, 798, and 788 represent threads for initiating in parallel communication using the various protocols, e.g., UDP, TCP, HTTP proxy, HTTP through port 80 (HTTP 80), and HTTP through port 8080 (HTTP 8080). Although only five protocol threads are shown, any number of protocol threads may be initiated by the client, using any conventional and/or suitable protocols. The steps associated with each of threads 794, 790, 792, 796, 798, and 788 are discussed herein in connection with Figs. 8A-8E and 9.

In Fig. 8A, the UDP protocol thread is executed. The client inquires in step 716 whether a UDP proxy is required. If the UDP proxy is required, the user may obtain the name of the UDP proxy from, for example, the system administrator in order to use the UDP proxy to facilitate communication to the proxy (in step 718). If no UDP proxy is required, the client may directly connect to the server (in step 720). Thereafter, the client may begin sending a data request (i.e., a control request) to the server in step 722 using the UDP protocol (either through the proxy if a proxy is involved or directly to the server if no proxy is required).

In Fig. 8B, the TCP protocol thread is executed. If TCP protocol is employed, the client typically directly connects to the server (in step 726). Thereafter, the client may begin sending a data request (i.e., a control request) to the server using the TCP protocol (step 724).

In Fig. 8C, the HTTP protocol thread is executed. The client inquires in step 716 whether an HTTP proxy is required. If the HTTP proxy is required, the user may obtain the name of the HTTP proxy from, for example, the browser since, as discussed earlier, the data pertaining to the proxy may be kept by the browser. Alternatively, the user may obtain data pertaining to the HTTP proxy from the system administrator in order to use the HTTP proxy to facilitate communication to the server (in step 732).

If no HTTP proxy is required, the client may directly connect to the server (in step 730). Thereafter, the client may begin sending a data request (i.e., a control request) to the server in step 734 using the HTTP protocol (either through the proxy if a proxy is involved or directly to the server if no proxy is required).

In Fig. 8D, the HTTP 80 protocol thread is executed. If HTTP 80 protocol is employed, HTTP data may be exchanged but only through port 80, which may be for example the port on the client computer through which communication with the network is permitted. Through port 80, the client typically directly connects to the server (in step 736). Thereafter, the client may begin sending a data request (i.e., a control request) to the server (step 738) using the HTTP 80 protocol.

In Fig. 8E, the HTTP 8080 protocol thread is executed. If HTTP 8080 protocol is employed, HTTP data may be exchanged but only through port 8080, which may be the port on the client computer for communicating with the network. Through port 8080, the client typically directly connects to the server (in step 740). Thereafter, the client may begin sending a data request (i.e., a control request) to the server (step 742) using the HTTP 8080 protocol. The multiplexing and demultiplexing techniques that may be employed for communication through port 8080, as well as port 80 of Fig. 8D, have been discussed earlier and are not repeated here for brevity sake.

Fig. 9 illustrates, in accordance with one embodiment of the present invention, control thread 794 of Fig. 7. It should be emphasized that Fig. 7 is but one way of implementing the control thread; other techniques of implementing the control thread to facilitate autodetect should be apparent to those skilled in the art in view of this disclosure. In step 746, the thread determines whether the predefined timeout period has expired. The predefined timeout period may be any predefined duration (such as 7 seconds for example) from the time the data request is sent out to the server (e.g., step 722 of Fig. 8A). In one embodiment, each protocol thread has its own timeout period whose expiration occurs at the expiration of a predefined duration after the data request using that protocol has been sent out. When all the timeout periods associated with all the protocols have been accounted for, the timeout period for the autodetect technique is deemed expired.

If the timeout has occurred, the thread moves to step 754 wherein the most advantageous protocol among the set of protocols received back from the server is selected for communication. As mentioned, the selection of the most advantageous protocol may be performed in accordance with some predefined priority scheme, and data regarding the selected protocol may be saved for future communication sessions between this server and this client.

If no timeout has occurred, the thread proceeds to step 748 to wait for either data from the server or the expiration of the timeout period. If timeout occurs, the thread moves to step 754, which has been discussed earlier. If data is received from the server, the thread moves to step 750 to ascertain whether the protocol associated with the data received from the server is the predefined "best" protocol, e.g., in accordance with the predefined priority.

If the predefined "best" protocol (e.g., UDP in some real-time data rendering applications) is received, the thread preferably moves to step 754 to terminate the autodetect and to immediately begin using this protocol for data communication instead of waiting for the timeout expiration. Advantageously, the duration of the autodetect sequence may be substantially shorter than the predefined timeout period. In this manner, rapid autodetect of the most suitable protocol and rapid establishment of communication are advantageously facilitated.

If the predefined "best" protocol is not received in step 750, the thread proceeds to step 752 to add the received protocol to the received set. This received protocol set represents the set of protocols from which the "most advantageous" (relatively speaking) protocol is selected. The most advantageous protocol is ascertained relative to other protocols in the received protocol set irrespective whether it is the predefined "best" protocol in accordance with the predefined priority. As an example of a predefined protocol priority, UDP may be deemed to be best (i.e., the predefined best), followed by TCP, HTTP, then HTTP 80 and HTTP 8080 (the last two may be equal in priority). As mentioned earlier, the most advantageous protocol is selected from the received protocol set preferably upon the expiration of the predefined timeout period.

From step 752, the thread returns to step 746 to test whether the timeout period has expired. If not, the thread continues along the steps discussed earlier.

Note that since the invention attempts to establish communication between the client application and the server computer in parallel, the time lag between the time the autodetect mechanism begins to execute and the time when the most advantageous protocol is determined is minimal. If communication attempts have been tried in serial, for example, the user would suffer the delay associated with each protocol thread in series, thereby disadvantageously lengthening the time period between communication attempt and successful establishment of communication.

The saving in time is even more dramatic in the event the network is congested or damaged. In some networks, it may take anywhere from 30 to 90 seconds before the client application realizes that an attempt to connect to the server (e.g., step 720, 726, 730, 736, or 740) has failed. If each protocol is tried in series, as is done in one embodiment, the delay may, in some cases, reach minutes before the user realizes that the network is unusable and attempts should be made at a later time.

By attempting to establish communication via the multiple protocols in parallel, network-related delays are suffered in parallel. Accordingly, the user does not have to wait for multiple attempts and failures before being able to ascertain that the network is unusable and an attempt to establish communication should be made at a later time. In one embodiment, once the user realizes that all parallel attempts to connect with the network and/or the proxies have failed, there is no need to make the user wait until the expiration of the timeout periods of each thread. In accordance with this embodiment, the user is advised to try again as soon as it is realized that parallel attempts to connect with the server have all failed. In this manner, less of the user's time is needed to establish optimal communication with a network.

While this invention has been described in terms of several preferred embodiments, there are alterations, permutations, and equivalents which fall within the scope of this invention. For example, although the invention has been described with reference to sending out protocol threads in parallel, the automatic protocol detection technique also applies when the protocol threads are sent serially. In this case, while it may take longer to select the most advantageous protocol for selection, the automatic protocol detection technique accomplishes the task without requiring any sophisticated technical knowledge on the part of the user of the client computer. The duration of the autodetect technique, even when serial autodetect is employed, may be shortened by trying the protocols in order of their desirability and ignoring less desirable protocols once a more desirable protocol is obtained. It should also be noted that there are many alternative ways of implementing the methods and apparatuses of the present invention. It is therefore intended that the following appended claims be interpreted as including all such alterations, permutations, and equivalents as fall within the true spirit and scope of the present invention.

What is claimed is:

1. In a computer network, a method for automatically detecting a most advantageous protocol for communication by a client computer, said client computer being configured to be coupled to a server computer via a computer network, comprising:

sending, from said client computer to said server computer, a plurality of data requests, each of said data requests employing a different protocol and a different connection, said data requests being configured to solicit, responsive to said data requests, a set of responses from said server computer, each of said responses employing a protocol associated with a respective one of said data requests;

receiving at said client computer at least a subset of said responses; and

monitoring said subset of said responses as each response is received from said server computer to select said most advantageous protocol from protocols associated with said subset of said responses, wherein said most advantageous protocol is determined based on a predefined protocol priority.

2. The method of claim 1 wherein said sending is performed in a substantially parallel manner.

3. The method of claim 2 wherein said plurality of data requests are sent substantially at the same time.

4. The method of claim 2 wherein said plurality of data requests execute concurrently.

5. The method of claim 2 wherein said most advantageous protocol represents a predefined best protocol, said client computer, upon receiving a response employing said predefined best protocol, immediately designates said predefined best protocol said most advantageous protocol.

6. The method of claim 3 wherein said control thread selects said most advantageous protocol upon an expiration of a timeout period if said predefined best protocol is not received at said client computer upon said expiration, said timeout period being measured from a transmitting time of said one of said data requests, said subset of responses representing responses received from said server computer during said timeout period.

7. The method of claim 4 wherein said client computer represents a computer executing an application for rendering real-time data as said real-time data is received from said server computer.

8. The method of claim 7 wherein said computer network represents the Internet and said predefined best protocol represents UDP.

9. The method of claim 8 wherein said real-time data represents one of a video data stream, an audio data stream, and an annotation data stream.

10. In a computer network, a method for automatically detecting a most advantageous protocol for communication by a client computer, said client computer being configured to be coupled to a server computer via a computer network, comprising:

sending, from said client computer to said server computer, a plurality of data requests, each of said data requests employing a different protocol and a different connection;

receiving at least a subset of said data requests at said server computer;

sending a set of responses from said server computer to said client computer, said set of responses being responsive to said subset of said data requests, each of said responses employing a protocol associated with a respective one of said subset of said data requests;

receiving at said client computer at least a subset of said responses; and selecting, for said communication between said client computer and said server computer, said most advantageous protocol from protocols associated with said subset of said responses, wherein said most advantageous protocol is determined based on a predefined protocol priority.

11. The method of claim 10 wherein said sending is performed in a substantially parallel manner.

12. The method of claim 10 wherein said most advantageous protocol represents a predefined best protocol, said selecting comprises:

monitoring, employing said client computer, said subset of said responses as each one of said subset of said responses is received at said client computer from said server computer for a response employing said predefined best protocol; and

designating said predefined best protocol said most advantageous protocol, thereby immediately permitting said client computer to employ said predefined best protocol for communication with said server computer without further receiving additional responses from said server computer.

13. The method of claim 10 wherein said selecting comprises:

selecting at an expiration of a timeout period said most advantageous protocol from said protocols associated with said subset of responses, said subset of responses representing responses received from said server computer during said timeout period.

14. The method of claim 13 wherein said timeout period represents a predefined time period associated with one of said data requests measured form a transmitting time of said one of said data requests.

15. The method of claim 14 wherein said client computer represents a computer executing an application for rendering real-time data as said real-time data is received from said server computer.

16. The method of claim 15 wherein said computer network represents the Internet and said predefined best protocol represents UDP.

17. The method of claim 15 wherein said real-time data represents one of a video data stream, an audio data stream, and an annotation data stream.

18. The method of claim 17 wherein said data requests employ at least one of a UDP, TCP, HTTP proxy, HTTP 80, and HTTP 8080 protocols.

19. The method of claim 18 wherein said HTTP 80 protocol represents a protocol for permitting multiple HTTP data streams to be transmitted in a multiplexed manner through port 80, said multiple HTTP data streams representing said video data stream and said audio data stream.

20. The method of claim 19 wherein said HTTP 8080 protocol represents a protocol for permitting multiple HTTP data streams to be transmitted in a multiplexed manner through port 8080, said multiple HTTP data streams representing said video data stream and said audio data stream.

21. A computer-readable medium containing computer-readable instructions for automatically detecting a most advantageous protocol for communication by a client computer, said client computer being configured for coupling to a server computer via a computer network, said computer-readable instructions comprise:

computer-readable instructions for sending in a substantially parallel manner, from said client computer to said server computer, a plurality of data requests, each of said data requests employing a different protocol and a different connection, said data requests being configured to solicit, responsive to said data requests, a set of responses from said server computer, each of said responses employing a protocol associated with a respective one of said data requests;

computer-readable instructions for receiving at said client computer at least a subset of said responses; and

computer-readable instructions for monitoring said subset of said responses as each response is received from said server computer to select said most advantageous protocol from protocols associated with said subset of said responses, wherein said most advantageous protocol is determined based on a predefined protocol priority.

22. The computer-readable medium of claim 21 wherein said most advantageous protocol represents a predefined best protocol, said client computer, upon receiving a response employing said predefined best protocol, immediately designates said predefined best protocol said most advantageous protocol.

23. The computer-readable medium of claim 22 wherein said control thread selects said most advantageous protocol upon an expiration of a timeout period if said predefined best protocol is not received at said client computer upon said expiration, said timeout period being measured from a transmitting time of said one of said data requests, said subset of responses representing responses received from said server computer during said timeout period.

24. The computer-readable medium of claim 23 wherein said client computer represents a computer executing an application for rendering real-time data as said real-time data is received from said server computer.

25. The computer-readable medium of claim 24 wherein said computer network represents the Internet and said predefined best protocol represents UDP.

26. The computer-readable medium of claim 25 wherein said real-time data represents one of a video data stream, an audio data stream, and an annotation data stream.

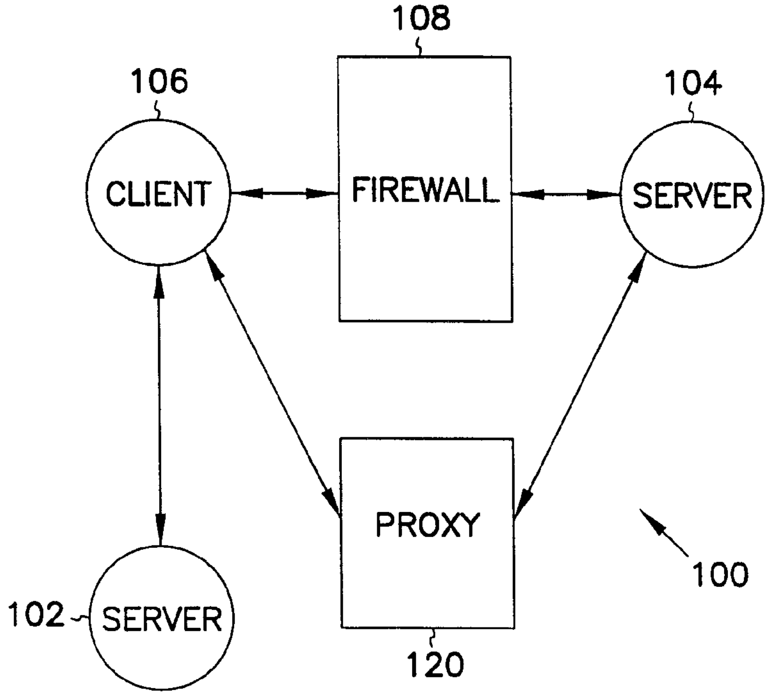


FIG. 1

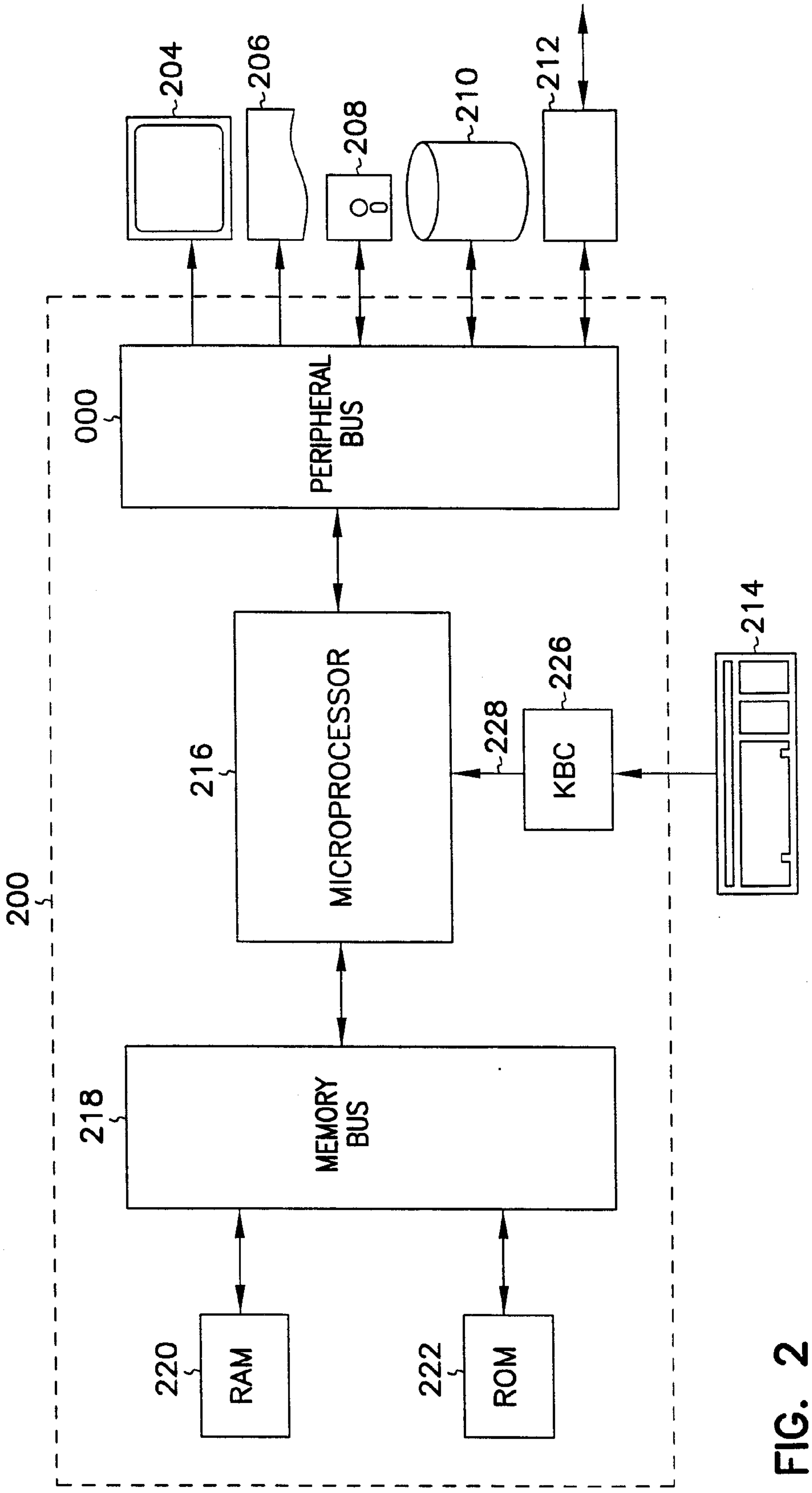

204
206
208
210
212
000
PERIPHERAL BUS
216
MICROPROCESSOR
228
226
KBC
214
200
218
MEMORY BUS
220
RAM
222
ROM

FIG. 2

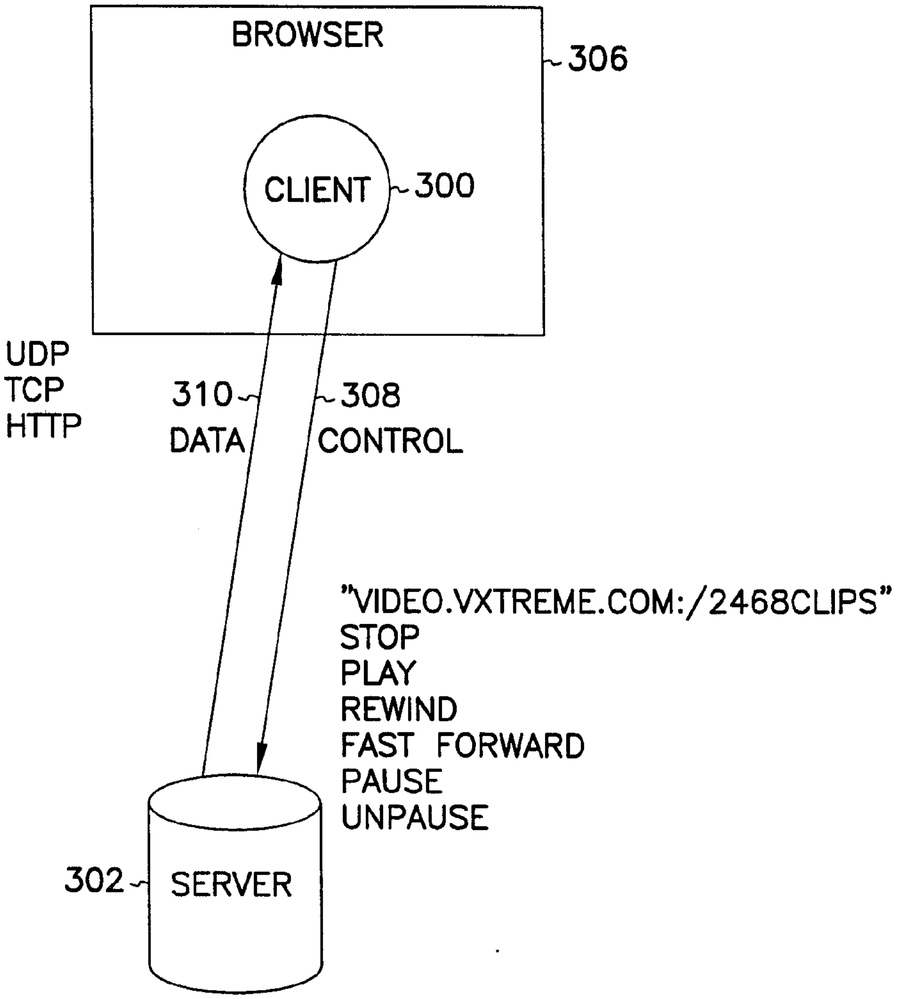

BROWSER
306
CLIENT
300
UDP
TCP
HTTP
310
DATA
308
CONTROL
"VIDEO.VXTREME.COM:/2468CLIPS"
STOP
PLAY
REWIND
FAST FORWARD
PAUSE
UNPAUSE
302
SERVER

FIG. 3

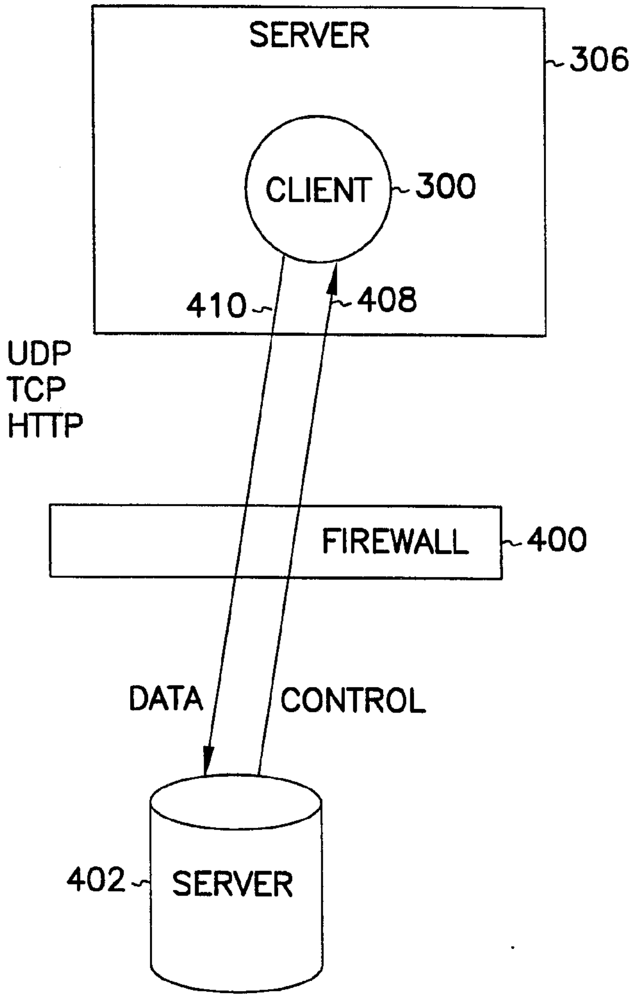

SERVER
306
CLIENT
300
410
408
UDP
TCP
HTTP
FIREWALL
400
DATA
CONTROL
402
SERVER

FIG. 4

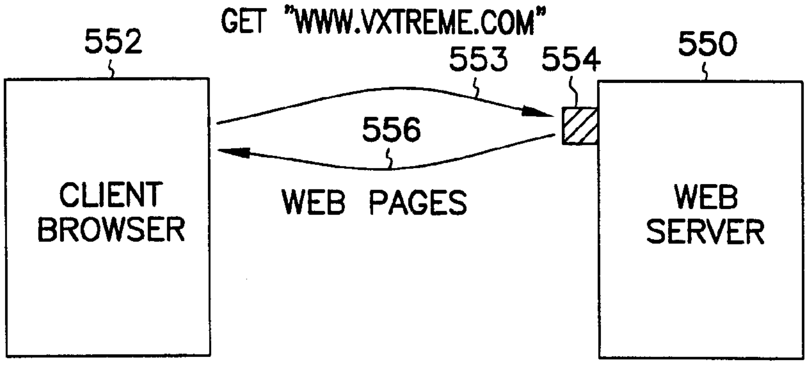


FIG. 5A (PRIOR ART)

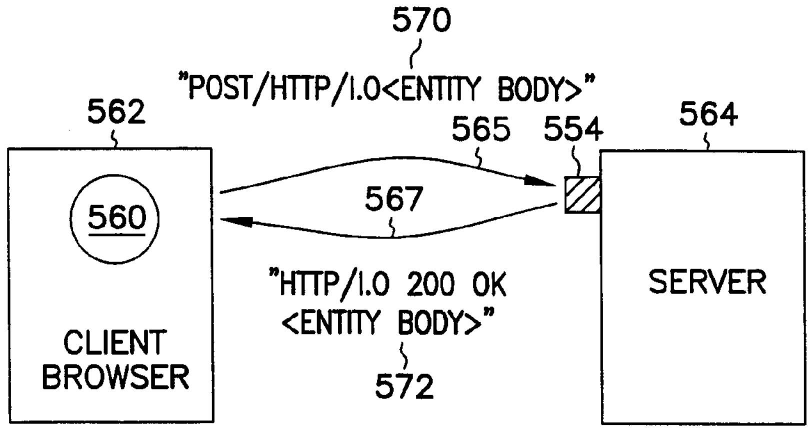


FIG. 5B

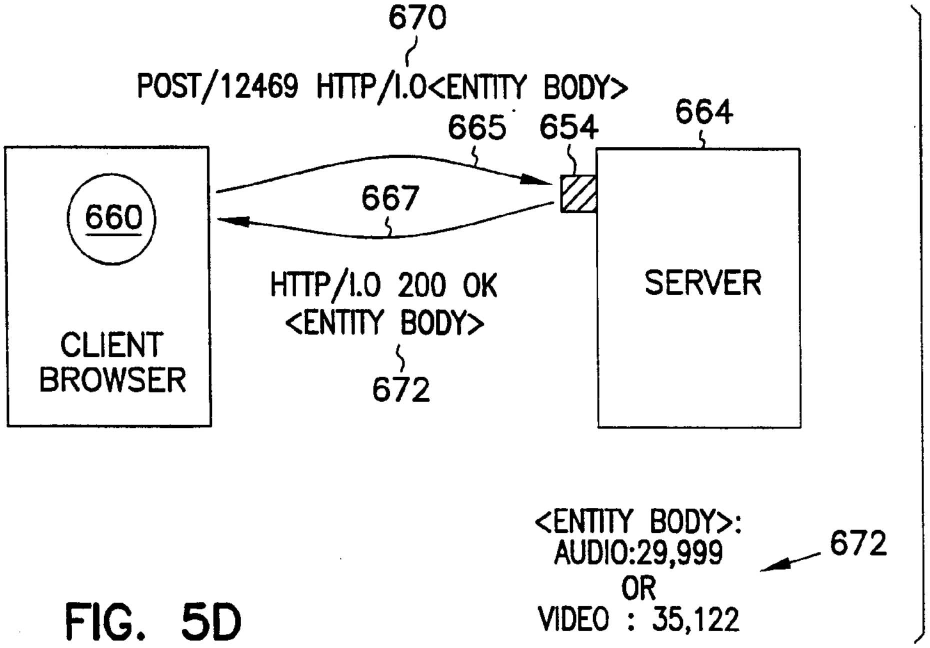

670
POST/12469 HTTP/1.0<ENTITY BODY>
665
654
664
660
667
CLIENT BROWSER
HTTP/1.0 200 OK <ENTITY BODY>
672
SERVER
<ENTITY BODY>: AUDIO:29,999 OR VIDEO : 35,122
672

FIG. 5D

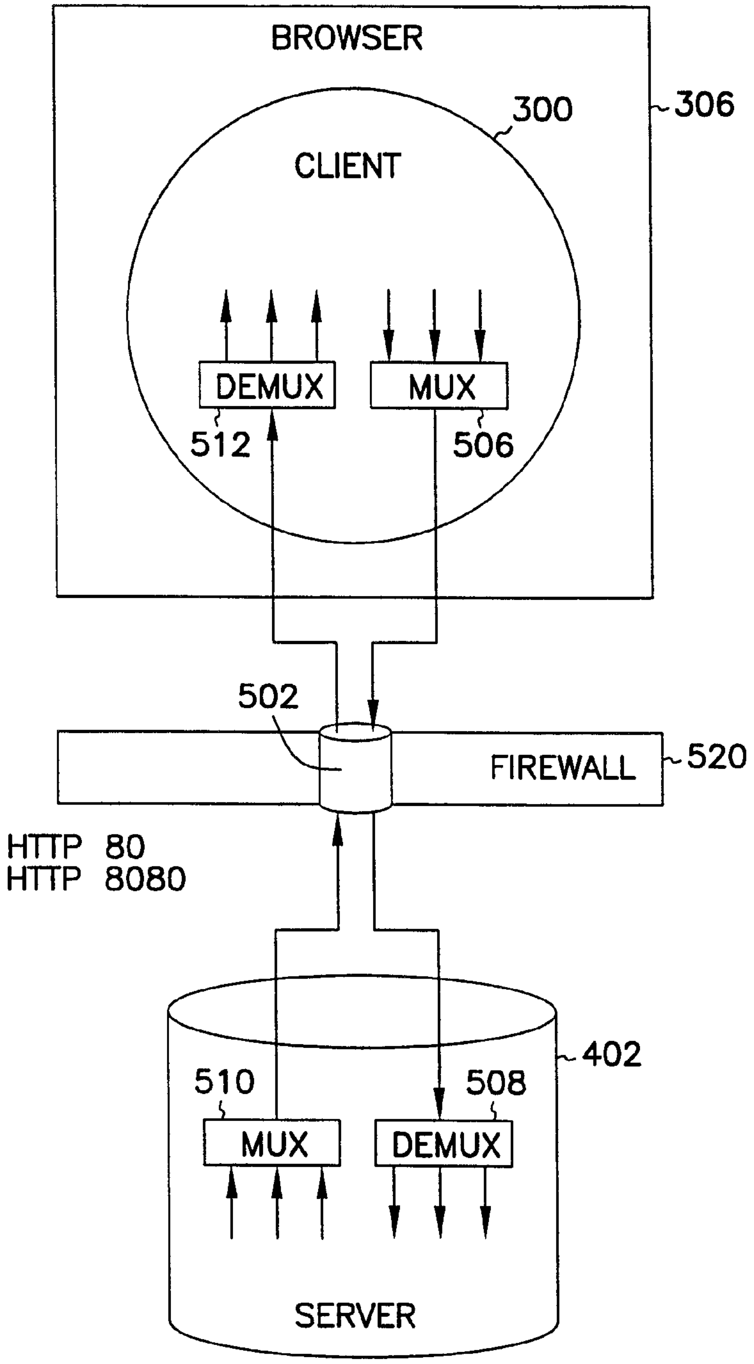

BROWSER
306
CLIENT
300
DEMUX
512
MUX
506
502
FIREWALL
520
HTTP 80
HTTP 8080
402
510
MUX
508
DEMUX
SERVER

FIG. 5C

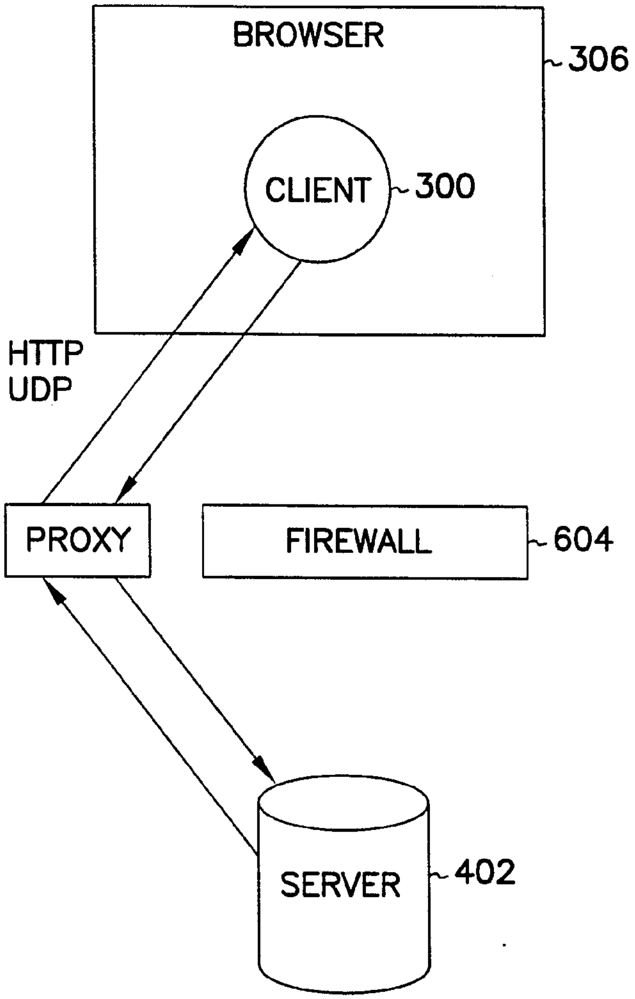

BROWSER
306
CLIENT
300
HTTP
UDP
PROXY
FIREWALL
604
SERVER
402

FIG. 6

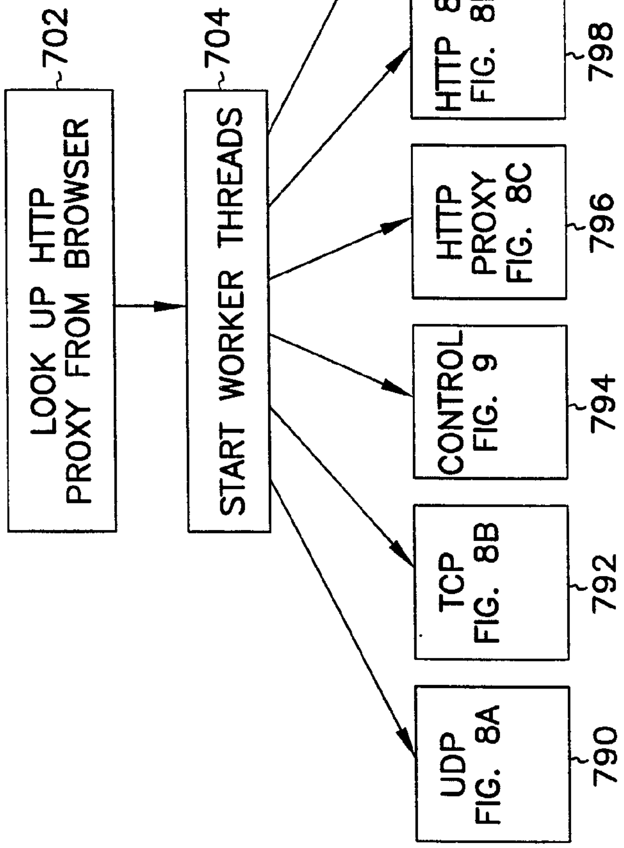

LOOK UP HTTP PROXY FROM BROWSER
702
START WORKER THREADS
704
UDP FIG. 8A
790
TCP FIG. 8B
792
CONTROL FIG. 9
794
HTTP PROXY FIG. 8C
796
HTTP 80 FIG. 8D
798
HTTP 8080 FIG. 8E
788

FIG. 7

FIG. 8A

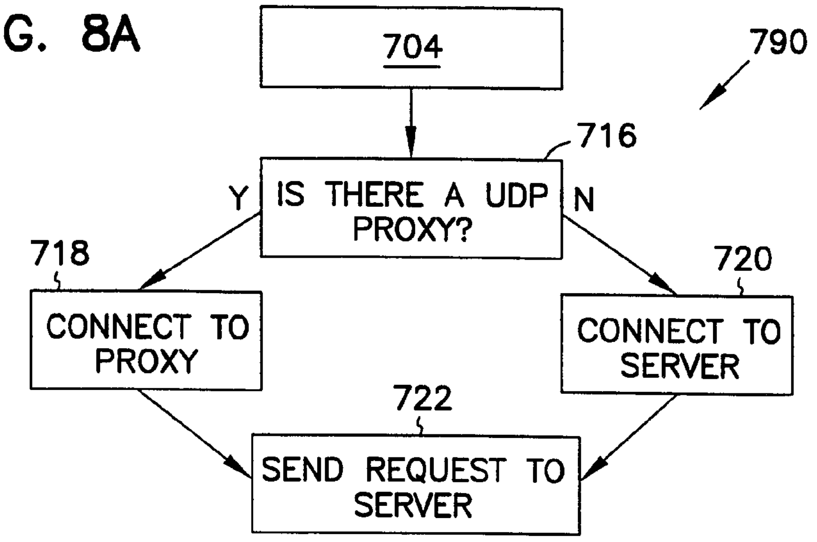


FIG. 8B

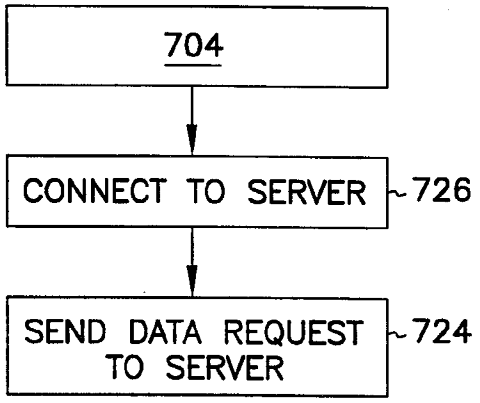


FIG. 8C

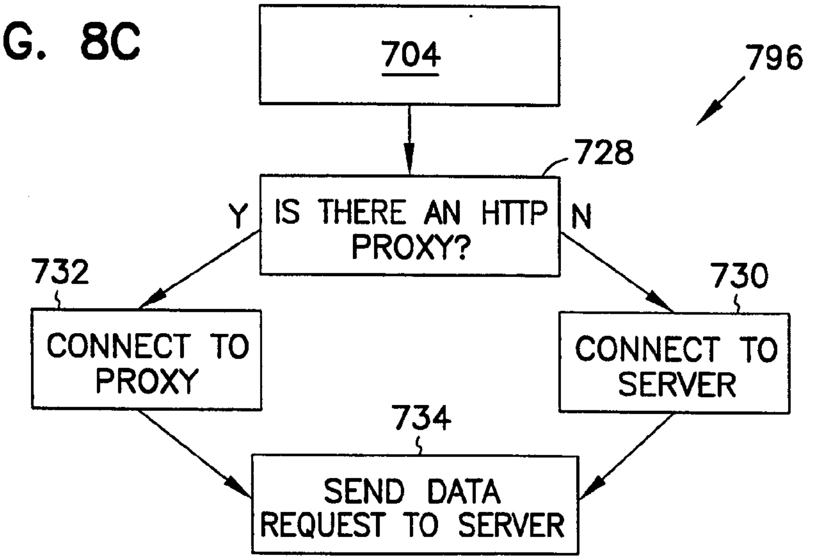


FIG. 8D

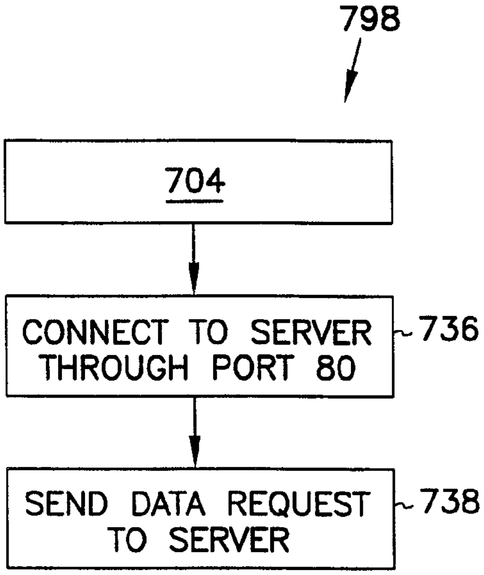
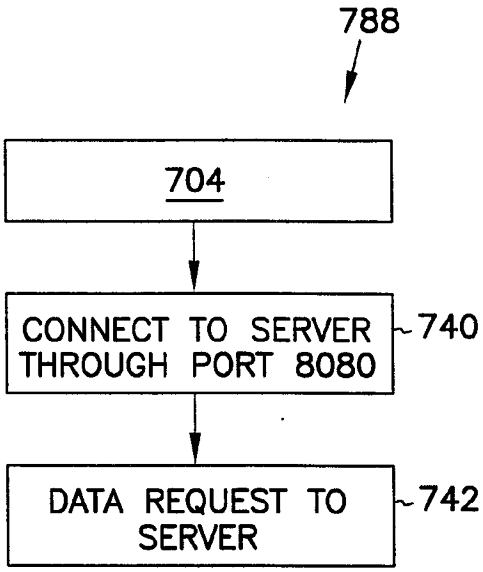

798
704
CONNECT TO SERVER THROUGH PORT 80
736
SEND DATA REQUEST TO SERVER
738

FIG. 8E


788
704
CONNECT TO SERVER THROUGH PORT 8080
740
DATA REQUEST TO SERVER
742

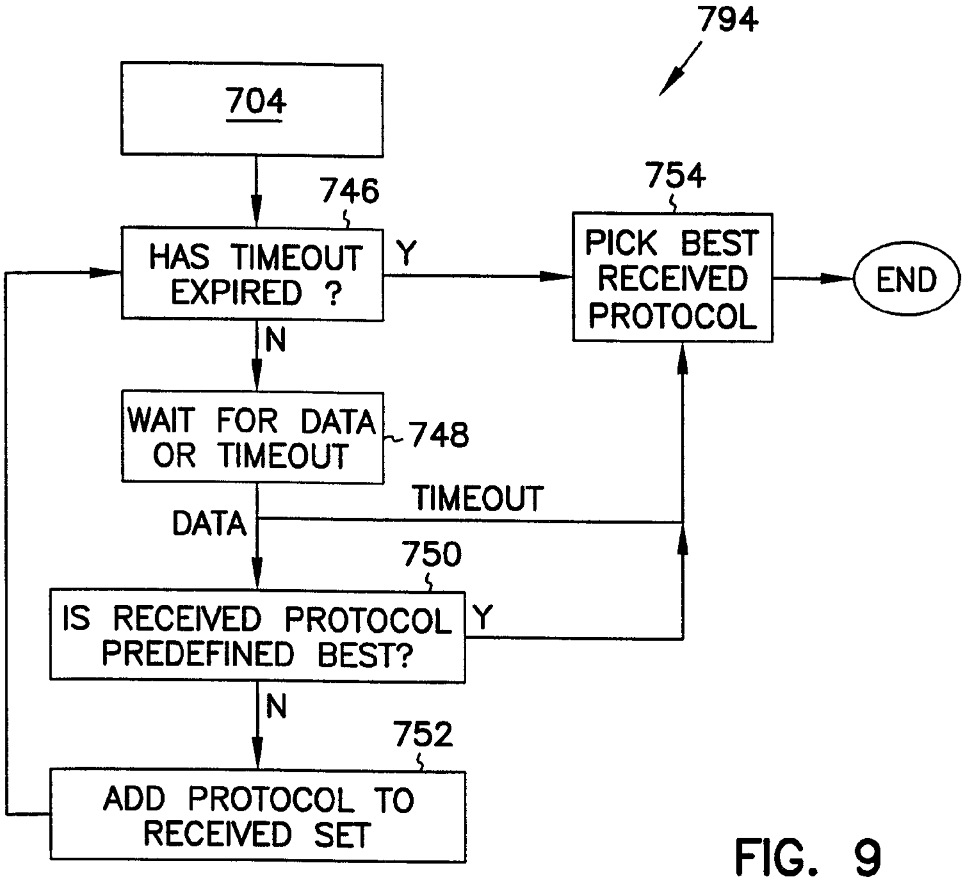

794
704
746
HAS TIMEOUT EXPIRED ?
Y
N
754
PICK BEST RECEIVED PROTOCOL
END
WAIT FOR DATA OR TIMEOUT
748
TIMEOUT
DATA
750
IS RECEIVED PROTOCOL PREDEFINED BEST?
Y
N
752
ADD PROTOCOL TO RECEIVED SET

FIG. 9

INTERNATIONAL SEARCH REPORT

Inte onal Application No
PCT/US 98/01797

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04L29/06 H04L29/08

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 613 274 A (IBM) 31 August 1994 see abstract see page 2, line 54 - line 55 see page 3, line 7 - line 11 ---	1,10,12
A	EP 0 751 656 A (CANON KK) 2 January 1997 see abstract see page 2, column 2, line 27 - line 40 see page 4, column 6, line 52 - line 57 see page 5, column 7, line 22 - column 8, line 13 see figure 2 see figure 3 see page 2, column 1, line 24 - line 27 see page 2, column 1, line 52 - column 2, line 2 -----	1-26

□ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of theinternational search 8 June 1998	Date of mailing of the international search report 17/06/1998
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Adkhis, F

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inte onal Application No
PCT/US 98/01797

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0613274 A	31-08-1994	US 5537417 A JP 7049823 A JP 8001622 B	16-07-1996 21-02-1995 10-01-1996
EP 0751656 A	02-01-1997	US 5710908 A JP 9062593 A	20-01-1998 07-03-1997

US006044401A

United States Patent [19]

Harvey

[11] **Patent Number: 6,044,401**

[45] **Date of Patent: Mar. 28, 2000**

[54] **NETWORK SNIFFER FOR MONITORING AND REPORTING NETWORK INFORMATION THAT IS NOT PRIVILEGED BEYOND A USER'S PRIVILEGE LEVEL**

[75] Inventor: **John Paul Harvey**, Round Rock, Tex.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **08/753,085**

[22] Filed: **Nov. 20, 1996**

[51] **Int. Cl.**^{7} **G06F 15/16**

[52] **U.S. Cl.** **709/225**; 713/201

[58] **Field of Search** 395/186, 187.01, 395/188.01, 200.53, 200.54, 200.55, 200.56; 709/223–226; 713/200–202

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,739,310	4/1988	Jeffers et al.	380/15
5,245,429	9/1993	Virginio et al.	358/142
5,333,308	7/1994	Ananthampillai	395/182.02
5,359,367	10/1994	Stockill	348/552
5,425,101	6/1995	Woo et al.	380/23
5,426,421	6/1995	Gray	395/200.53
5,519,780	5/1996	Woo et al.	380/49
5,610,905	3/1997	Murthy et al.	370/401
5,648,965	7/1997	Thadani et al.	370/241
5,798,706	8/1998	Kraemer et al.	340/825.07
5,828,846	10/1998	Kirby et al.	709/238
5,933,602	8/1999	Grover	709/224
5,950,195	9/1999	Stockwell et al.	707/4
5,956,715	9/1999	Glasser et al.	707/9

OTHER PUBLICATIONS

Jacobson et al., "tcpdump", manual included with tcpdump program, pp. 1–26, Jun. 20, 1994.

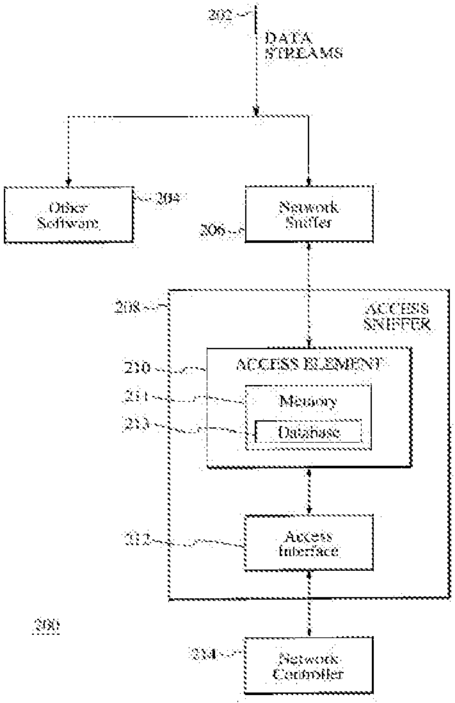
"A Very Brief Guide to tcpdump", Internet, pp. 1–2, Sep. 19, 1996.

Primary Examiner—Zarni Maung
Assistant Examiner—Andrew Caldwell
Attorney, Agent, or Firm—Sawyer & Associates; Volel Emile

[57] **ABSTRACT**

The present invention provides a method and system for locating available information in a network environment by a user in a node. In a system aspect, within a node in the network, the present invention includes a network sniffer and an access sniffer. The access sniffer includes an access element and an access interface. The access element preferably includes a memory and a database. The access element accesses the network sniffer and filters out unavailable information by using information such as address and port numbers gathered by the network sniffer. Unavailable information includes information which is non-public or beyond the privilege level of the particular user. The access element evaluates data streams which are public information to determine if the data streams meet a predetermined criteria. If the data streams meet the predetermined criteria, then the data is saved in the database. The access element transfers only the information available to the particular user to the access interface. The access element can time itself for a limited amount of time for execution. Once the predetermined time period has expired, the access element is complete and it can save and transfer the appropriate information to the access interface.

19 Claims, 3 Drawing Sheets



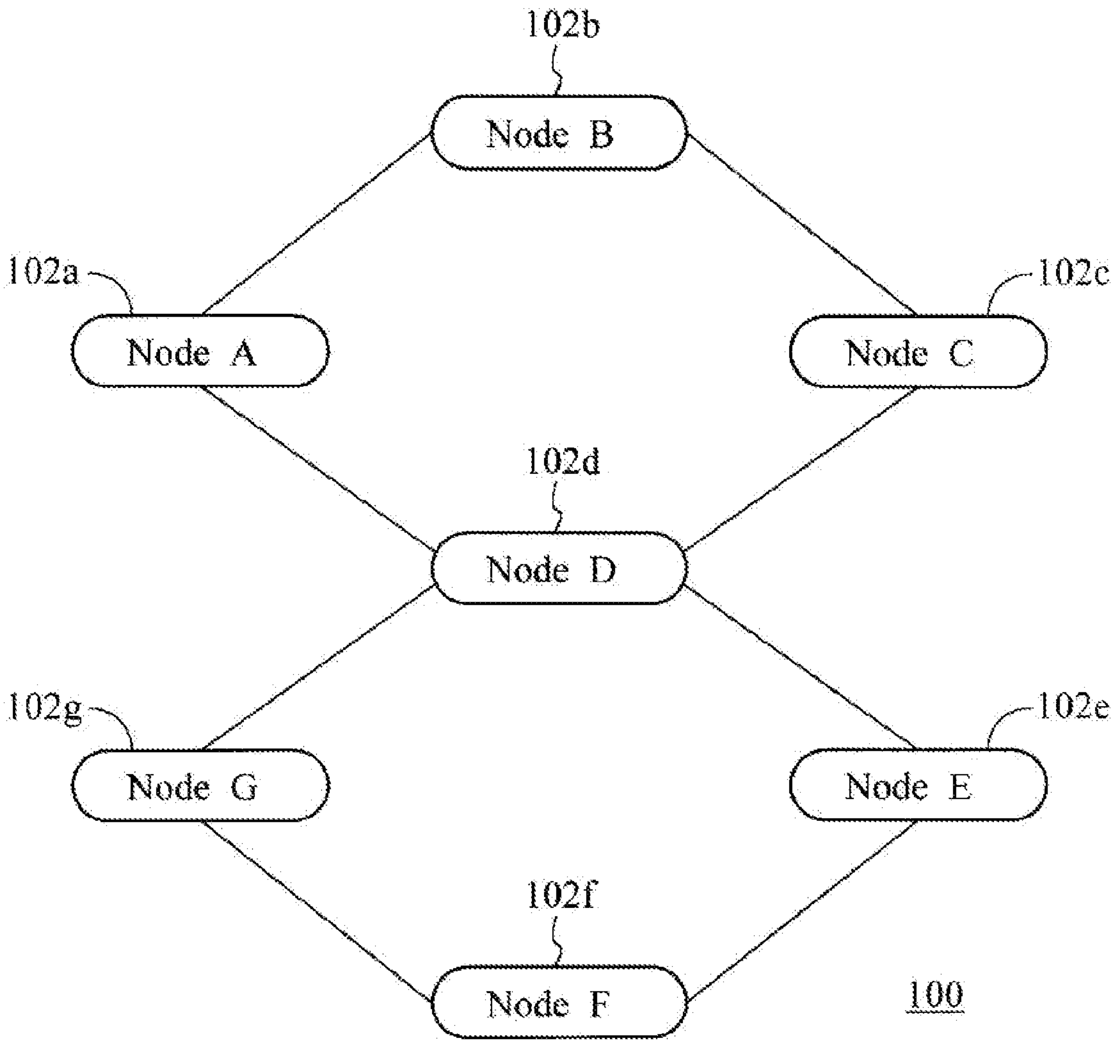


FIG. 1
(PRIOR ART)

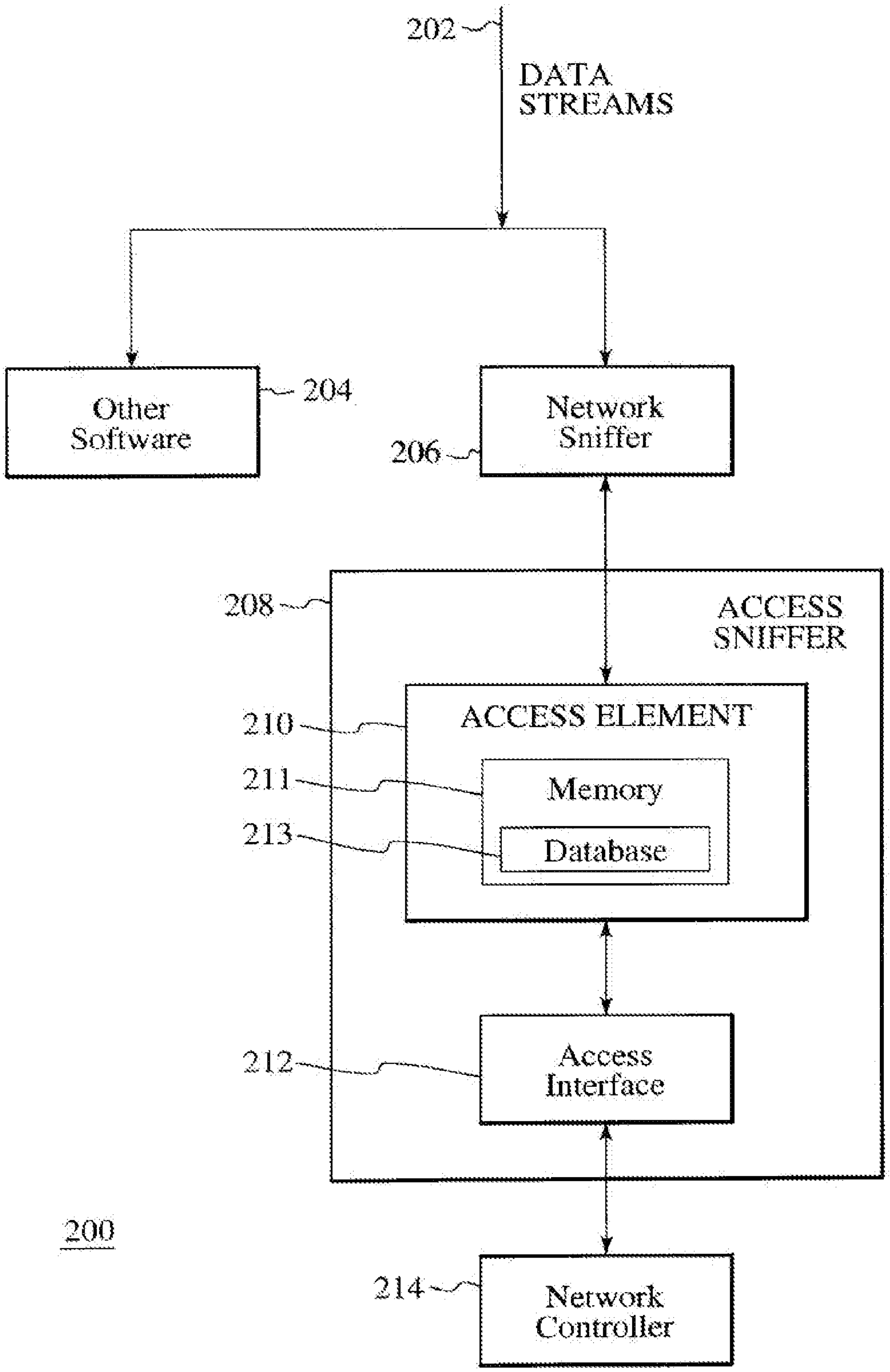

202
DATA STREAMS
Other Software
204
Network Sniffer
206
208
ACCESS SNIFFER
210
ACCESS ELEMENT
211
Memory
213
Database
212
Access Interface
200
214
Network Controller

FIG. 2

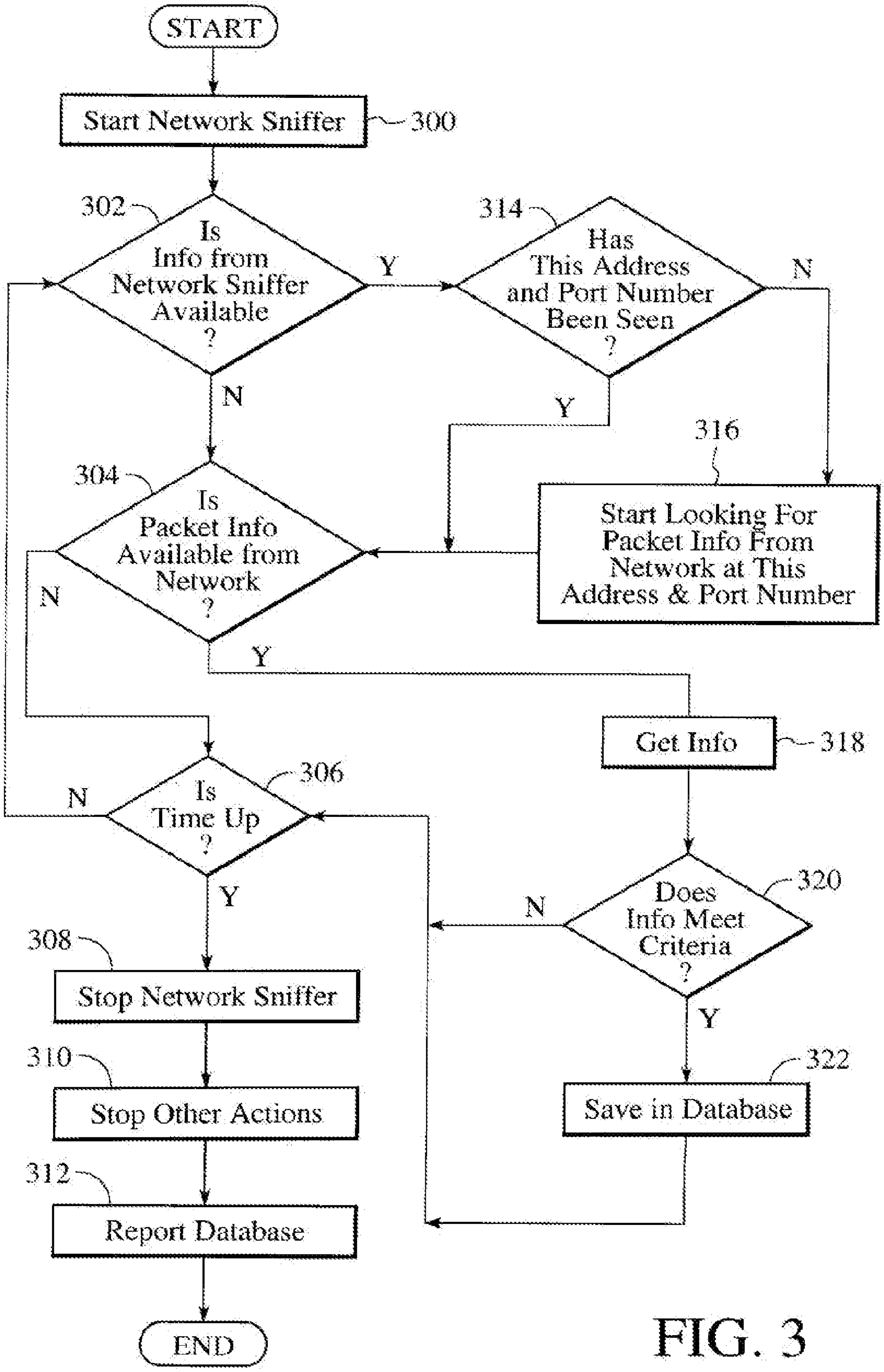

START
Start Network Sniffer
300
302
Is Info from Network Sniffer Available ?
Y
N
314
Has This Address and Port Number Been Seen ?
N
Y
316
Start Looking For Packet Info From Network at This Address & Port Number
304
Is Packet Info Available from Network ?
N
Y
Get Info
318
N
Is Time Up ?
306
Y
320
Does Info Meet Criteria ?
N
Y
308
Stop Network Sniffer
310
Stop Other Actions
312
Report Database
322
Save in Database
END

FIG. 3

NETWORK SNIFFER FOR MONITORING AND REPORTING NETWORK INFORMATION THAT IS NOT PRIVILEGED BEYOND A USER'S PRIVILEGE LEVEL

TECHNICAL FIELD

The present invention is related to locating information in a network environment, particularly available information in a network environment being located by a node in the network through the use of a network sniffer along with an access sniffer.

BACKGROUND

A network environment typically includes multiple nodes in which a node itself can also be a network. Information can be transmitted from one of these nodes which can be received by another node in the network.

Multimedia data streams can be sent through public networks for reception by the general public. The data streams can consist of audio, video, whiteboard, or any other type of digital data. A user on the network can receive these data streams using the appropriate software and with knowledge of the multicast address and port number for the stream. The problem is that it is difficult to locate the desired information since the user must typically know the address and port number for a desired data stream.

A network sniffer is a system and method that is normally used to monitor network activity when resolving network problems or when improving network efficiency. Although the network sniffer can typically access information regarding all data being transferred into the node, access to the network sniffer is generally privileged due to the sensitivity of some of the data on the networks. Thus, there is a need to facilitate ease of access to available information in a network environment. The present invention addresses such a need.

SUMMARY

The present invention is a system and method for locating requested data streams by accessing data monitored by a network sniffer for a particular node and evaluating data which is within the privilege level of a particular user.

A network sniffer is a system and method that is normally used to monitor network activity when resolving network problems or when improving network efficiency. Because of the sensitivity of some of the data on networks, average users are not generally allowed to access network sniffers.

A method according to the present invention for locating available information in a network environment by a node in the network comprising the steps of accessing a network sniffer which can monitor data sent to a node; determining if the data being monitored by the sniffer is privileged; and accessing data which is not privileged beyond the user's level.

The present invention can locate an address and port number of a data packet through the use of the network sniffer, determine if it has already been processed, evaluate the data packet associated with the address and port number, and determine if it meets predetermined criteria. If it does meet the predetermined criteria, then the data is saved in a database.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram representing a network.

FIG. 2 is a functional block diagram of a system according to the present invention.

FIG. 3 is a flow diagram of a method according to the present invention.

DETAILED DESCRIPTION

The illustrative embodiment is related to a system and method for locating available information in a network. The following description is presented to enable on of ordinary skill in the art to make and use the illustrative embodiment and is provided in the context of a patent application and its requirements. Various modifications to the illustrative embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. Thus, the illustrative embodiment is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

FIG. 1 is a diagram depicting a network environment **100**. The network **100** includes multiple nodes **102***a*–**102***d*, in which a node itself can also be a network, as shown by **102***d*–**102***g*. Information, such as video and audio, can be transmitted from one of the nodes such as the node **102***a*. This information can be received by another node in the network such as node **102***f* by using appropriate software and the knowledge of the multicast address and port number for the data stream.

The present invention uses services available through a network sniffer to examine the network connections and list the addresses and port numbers of available data streams such as multicast streams.

The present invention allows non-privileged users to access a network sniffer for gathering public information. It can also allow users to access a network sniffer for gathering information which is available up to the particular privilege level of the user. Public information and information which is available up to the particular privilege level of the user is herein collectively referred to as "available information." Non-public and sensitive information beyond the privilege level of the user is not reported. The available information may be further filtered so that only data meeting predetermined criteria, such as data for particular applications, are reported.

There are many ways in which a piece of information may be deemed privileged. Privileged information can include, but is not limited to, the following:

(1) a packet of information form the network having a destination address that is different from the address of the node the user is using;

(2) a packet of information having a source or destination port number that is reserved for privileged processes;

(3) a packet of information having a source or destination port number that is well known for passing privileged information;

(4) a packet of information which information is scrambled or encrypted;

(5) a packet of information which is not scrambled or encrypted, but contains a recognized structure that indicates that the data is privileged;

(6) a packet of information which is not scrambled or encrypted, but contains a recognized structure that indicates that the destination is some other user.

There are also many ways in which a piece of information may be deemed not privileged or not beyond the particular user's privilege level. Information which is not privileged can include, but is not limited to, the following:

(1) a packet of information which is not encrypted and contains a recognized structure that indicates that the data is not privileged;

(2) a packet of information having a source or destination port number that is well known for passing non-privileged information.

In addition to including the information which is not privileged, information which is not beyond the particular user's privilege level can also include, but is not limited to, the following:

(1) a packet of information which is scrambled or encrypted and can be unscrambled or decrypted with information (a key) provided by the user.

(2) a packet of information which is not scrambled or encrypted, but contains a recognized structure that indicates that the destination is the particular user.

Although the examples described herein are for multicast data, the method and system of the present invention also applies to broadcast, unicast, and other data that can be identified as available to the user.

FIG. **2** is a functional block diagram of a system according to the present invention. The basic components of system **200**, which is preferably located within a node in a network, include a network sniffer **206**, an access sniffer **208**, and a network controller **214**. The access sniffer **208** includes access element **210** and access interface **212**. The access element **210** preferably includes a memory **211** and a database **213**.

Data streams **202** enter the node and are utilized by software **204** in the system **200**. Copies of the data streams **202**, or portions thereof, can be accessed by the network sniffer **206**. The access sniffer **208** can then access the data streams being evaluated by the network sniffer **206**. The access element **210** accesses the network sniffer **206** and filters out unavailable information by using information such as address and port numbers gathered by the network sniffer **206**. Unavailable information includes information which is non-public or beyond the privilege level of the particular user.

The access element **210** can evaluate data streams which are public information to determine if the data streams meet a predetermined criteria, such as multicast real time protocol (RTP), for video and audio. Examples of addresses which the access elements **210** can utilize include multicast and unicast addresses.

Access element **210** preferably removes all unacceptable data streams. It preferably executes virtually all the major functions of the access sniffer **208**. The access element **210** can exclude data streams which are not targeted for this particular node, and exclude addresses destined for other software on the node. It can also execute unicast addresses if one of the predetermined criteria is to look for a multicast addresses. It can run tests on the data streams to determine if the data streams are valid. If it is not valid, then the access element **210** can exclude it. If it is valid, then it can determine whether the address and port number associated with that valid data stream are information available to the particular user. Access element **210** can also monitor the port number for data streams which meet the predetermined criteria.

The access element **210** can also check to see if a particular address and port number have been evaluated in the past. If so, then it ignores the previously evaluated address and port number. If the address and port number have not been previously evaluated, and they also meet the predetermined criteria, then these data streams are kept in the data base **213**. Examples of predetermined criteria include a particular privilege level such as public information, a type of information such as a particular node name, type name, or user name of the particular source.

Once access element **210** finds data streams which meet the predetermined criteria, it can then save that information along with the port number and the address associated with it. When the access element **210** has completed its functions, it preferably saves the filtered data and transfers only the information available to the particular user to the access interface **212**. It is preferable that the access element **210** does not transfer encrypted information or password information to the access interface **212**.

The access element **210** can time itself for a limited amount of time for execution. Once the predetermined time period has expired, the access element **210** is complete, and it can save and transfer the appropriate information to the access interface **212**.

One example of how the access element **210** maintains information in the database **213** is to open what is referred to in UNIX as a socket. The access element **210** can obtain a body of a data packet from the socket by looking for a specific port number and address. If the data stream located in the data base **213** is a particular type of data stream, such as a type name, a particular node name, or user name of the particular source, then that type can also be added to the data base **213**.

At the end of the predetermined time, the access element **210** preferably stops the process, closes all the sockets it opened, evaluates the database **213**, converts the requested information to a string, and passes it to the access interface **212**.

The information in the network sniffer **206** and the access element **210** are preferably privileged information, thus the user preferably does not have direct access to the access element **210**, but rather utilizes the access interface **212** to access only the information which is either public or within that particular user's privilege level. The information within the access interface **212** can then be transferred to the network controller **214**. Although the access interface is shown in FIG. **2** to be included in the access sniffer **208**, the access interface **212** can be a separate component and the access element **210** can then function as the access sniffer **208**.

FIG. **3** is a flow diagram of a method according to the present invention. The method shown in FIG. **3** is merely one example of obtaining the desired results according to the present invention. In this example, the network sniffer is started via step **300**. It is then determined if the information from the network sniffer is available via step **302**. The access element **210** of FIG. **2** would preferably perform this determination. As previously stated, available information means either public information or information which is within the particular user's privilege level. If the information from the network sniffer is not available, then it is determined, preferably by access element **210** of FIG. **2**, if the packet information is available form the network via step **304**. Note that the order of step **304** and **302** is completely interchangeable without affecting the results.

If the information form the data packet is not available, then it is determined if the predetermined time has elapsed via step **306**. If the predetermined time has not elapsed, then another data packet is evaluated by the access element **210** from the network sniffer via step **302** and **304**. If the predetermined time has elapsed, then the network sniffer is stopped via step **308** and any other actions which are in progress are also stopped via step **310**. The other actions include actions via steps **314**-**322** which would also preferably be performed by access element **210** of FIG. **2**. Information in the database is then reported to the access interface **212** of FIG. **2** via step **312**.

If information from the network sniffer is available via step **302**, then it is determined whether the address and port number of the data stream have previously been evaluated via step **314**. If it has been previously evaluated, then it is determined if the packet information is available from the network via step **304**. If it has not been previously evaluated, then the packet information from the network is looked for at the particular address and port number via step **316**. Then it is determined if the packet information is available from the network via step **304**. If the packet information is available from the network to the particular user via step **304**, then the information is retrieved via step **318**. It is then determined if the information meets the predetermined criteria via step **320**. If it does not, then it is determined whether the predetermined time has elapsed via step **306**. If, however, the information does meet the predetermined criteria, then it is saved in the data base via step **322**. Again, it is determined if the predetermined time has elapsed via step **306**. If so, then steps **308**–**312** are executed. If the predetermined time has not elapsed, then the next piece of information from the network sniffer is evaluated via step **302**.

Although the system and method has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations to the embodiments and those variations would be within the spirit and scope of the above-described system and method. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.

What is claimed is:

1. A method for locating available information in a network environment by a user at a node in the network, the method comprising the steps of:

a) accessing the network sniffer that is monitoring data sent to the node;

b) determining if the data being monitored by the network sniffer is privileged; and

c) accessing data which is not privileged beyond the user's privileged level.

2. A method for locating at a node requested data from among data sent to the node in a network environment, the requested data being located for providing access thereto by a user at the node, the method comprising the steps of:

a) accessing a network sniffer that is monitoring the data sent to the node;

b) determining if the data being monitored by the network sniffer is privileged or public;

c) determining if the data being monitored includes the available information if the data is privileged, wherein a privilege level of the available information is at or below a privilege level of the user;

d) locating addresses and port numbers of the available information, wherein the available information from among the data being monitored includes the data which is public;

e) reporting the available information including the corresponding addresses and port numbers; and

f) providing access to the available information, wherein the available information includes the requested data and wherein the requested data is not privileged beyond the user's privilege level.

3. The method of claim **2**, wherein the providing access step (f) further includes the step (f1) of transferring the

requested data including the corresponding addresses and port numbers to the user at the node.

4. The method of claim **3** wherein the data determining step (b) further includes step (b**2**) of determining if a particular address and port number have already been processed.

5. The method of claim **3**, wherein the data determining step (b) further includes step (b2) of evaluating a data packet associated with the address and port number.

6. The method of claim **2**, wherein the reporting step (e) further includes the step (e**1**) of determining if the requested data which meets a predetermined criteria is included in the available information by determining if the data being monitored which comprises the available information meets the predetermined criteria.

7. The method of claim **6**, wherein the reporting step (e) further includes the step (e**2**) of storing the requested data in a database if the data being monitored, which comprises the available information, meets the predetermined criteria.

8. The method of claim **6**, wherein the predetermined criteria includes a data valid requirement.

9. The method of claim **6**, wherein the requested data includes data which is audio data.

10. The method of claim **6**, wherein the requested data includes data which is video data.

11. The method of claim **2**, further comprising the step (g) of determining if time is up for stopping the accessing of the sniffer in step (a) after a predetermined time period and, if so, terminating all actions (steps a–f) including accessing the sniffer.

12. A system for locating available information in the network environment by a user in a node, the system comprising:

means for monitoring data which is sent to the node; and

means coupled to the monitoring means for accessing the data, wherein the accessing means determines if the data being monitored by the monitoring means is privileged beyond the user's privilege level, and wherein the accessing means reports the data which is not privileged beyond the user's privilege level and, with the data, the accessing means reports addresses and port numbers corresponding thereto.

13. The system of claim **12**, further comprising a memory means coupled to the accessing means for storing the data being accessed if the accessed data meets a predetermined criteria.

14. The system of claim **12**, further comprising an interface means for transferring the data which is not privileged beyond the user's privilege level.

15. A system for locating available information in the network environment by user data node, the system comprising:

a first sniffer for monitoring data which is sent to a node; and

a second sniffer coupled to the first sniffer for accessing the data which is being monitored by the first sniffer, wherein the second sniffer determines if the data being monitored is privileged beyond the user's privilege level, and wherein the accessing means reports the data which is not privileged beyond the user's privilege level and, with the data, the accessing means reports addresses and port numbers corresponding thereto.

16. The system of claim **15**, wherein the second sniffer further comprises a memory for storing accessed data if the accessed data meets a predetermined criteria.

17. The system of claim **15**, further comprising an interface for transferring the data which is not privileged beyond the user's privilege level.

18. A system for locating available information in a network environment by a user edit node, the system comprising:

accessing means for accessing data which is being monitored by a monitoring means, wherein the accessing means determines is the data being monitored is privileged beyond the user's privilege level, and wherein the accessing means reports the data which is not privileged beyond the user's privileged level and, with the data, the accessing means reports addresses and port numbers corresponding thereto; and

interface means coupled to the accessing means for transferring accessed data which is not beyond the user's privilege level.

19. A computer readable medium containing program instructions for locating at a node requesting data from among data sent to the node in and network environment, the requested data being located for providing access thereto by a user at a node, the program instructions for:

a) accessing the networks never that is monitoring the data sent to the node;

b) determining if the data being monitored by the network sniffers is privileged or public;

c) determining if it data being monitored includes the available information if the data is privileged, wherein a privilege level of the available information is at or below a privilege level of the user;

d) locating addresses and port numbers of the available information wherein the available information from among the data being monitored includes the data which is public;

e) reporting the available information including the corresponding addresses and port numbers; and

f) providing access to the available information, wherein the available information includes the requested data and wherein the requested data is not privileged beyond the user's privilege level.

* * * * *

(G) Art submitted by Applicant - Entered

The following items (1) – (4) listed below are hereby entered as evidence submitted to the Examiner as noted on an IDS transmission received by the USPTO on October 01, 2002.

(1) Copy of U.S. Patent Number 6,044,401 ("Harvey"). This evidence was entered into the record by the Applicant on 09/26/2002.

(2) Copy of WO 98/34385 A1 ("Vellanki et al."). This evidence was entered into the record by the Applicant on 09/26/2002.

(3) Copy of EP 0 612 274 A2 ("IBM"). This evidence was entered into the record by the Applicant on 09/26/2002.

(4) Copy of NPL - COOPER, I, et al., Web Proxy Auto-Discovery Protocol, Internet Online, 11/15/2000 URL: http://www.wrec.org/Drafts/draft-cooper-webi-wpad-00.txt XP-002209978. This evidence was entered into the record by the Applicant on 09/26/2002.

Copies of all References follows.

//

US006044401A

United States Patent [19]

Harvey

[11] **Patent Number: 6,044,401**

[45] **Date of Patent: Mar. 28, 2000**

[54] **NETWORK SNIFFER FOR MONITORING AND REPORTING NETWORK INFORMATION THAT IS NOT PRIVILEGED BEYOND A USER'S PRIVILEGE LEVEL**

[75] Inventor: **John Paul Harvey**, Round Rock, Tex.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **08/753,085**

[22] Filed: **Nov. 20, 1996**

[51] **Int. Cl.**^{7} **G06F 15/16**

[52] **U.S. Cl.** **709/225**; 713/201

[58] **Field of Search** 395/186, 187.01, 395/188.01, 200.53, 200.54, 200.55, 200.56; 709/223–226; 713/200–202

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,739,510	4/1988	Jeffers et al.	380/15
5,245,429	9/1993	Virginio et al.	358/142
5,333,308	7/1994	Ananthampillai	395/182.02
5,359,367	10/1994	Stockill	348/552
5,425,101	6/1995	Woo et al.	380/23
5,426,421	6/1995	Gray	395/200.53
5,519,780	5/1996	Woo et al.	380/49
5,610,905	3/1997	Murthy et al.	370/401
5,648,965	7/1997	Thadani et al.	370/241
5,798,706	8/1998	Kraemer et al.	340/825.07
5,828,846	10/1998	Kirby et al.	709/238
5,933,602	8/1999	Grover	709/224
5,950,195	9/1999	Stockwell et al.	707/4
5,956,715	9/1999	Glasser et al.	707/9

OTHER PUBLICATIONS

Jacobson et al., "tcpdump", manual included with tcpdump program, pp. 1–26, Jun. 20, 1994.

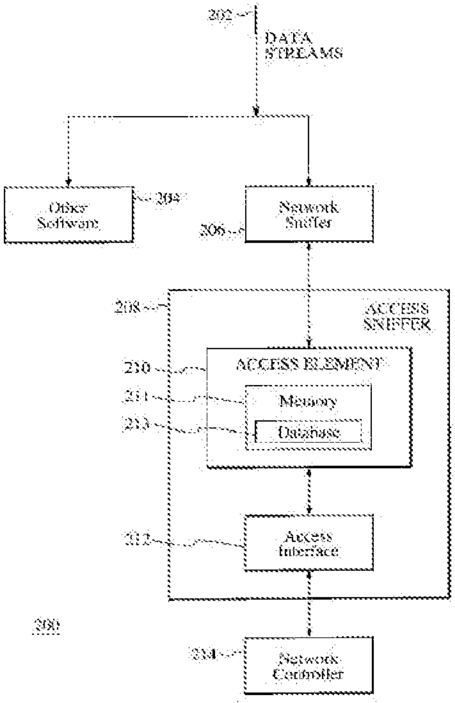
"A Very Brief Guide to tcpdump", Internet, pp. 1–2, Sep. 19, 1996.

Primary Examiner—Zarni Maung
Assistant Examiner—Andrew Caldwell
Attorney, Agent, or Firm—Sawyer & Associates; Volel Emile

[57] **ABSTRACT**

The present invention provides a method and system for locating available information in a network environment by a user in a node. In a system aspect, within a node in the network, the present invention includes a network sniffer and an access sniffer. The access sniffer includes an access element and an access interface. The access element preferably includes a memory and a database. The access element accesses the network sniffer and filters out unavailable information by using information such as address and port numbers gathered by the network sniffer. Unavailable information includes information which is non-public or beyond the privilege level of the particular user. The access element evaluates data streams which are public information to determine if the data streams meet a predetermined criteria. If the data streams meet the predetermined criteria, then the data is saved in the database. The access element transfers only the information available to the particular user to the access interface. The access element can time itself for a limited amount of time for execution. Once the predetermined time period has expired, the access element is complete and it can save and transfer the appropriate information to the access interface.

19 Claims, 3 Drawing Sheets



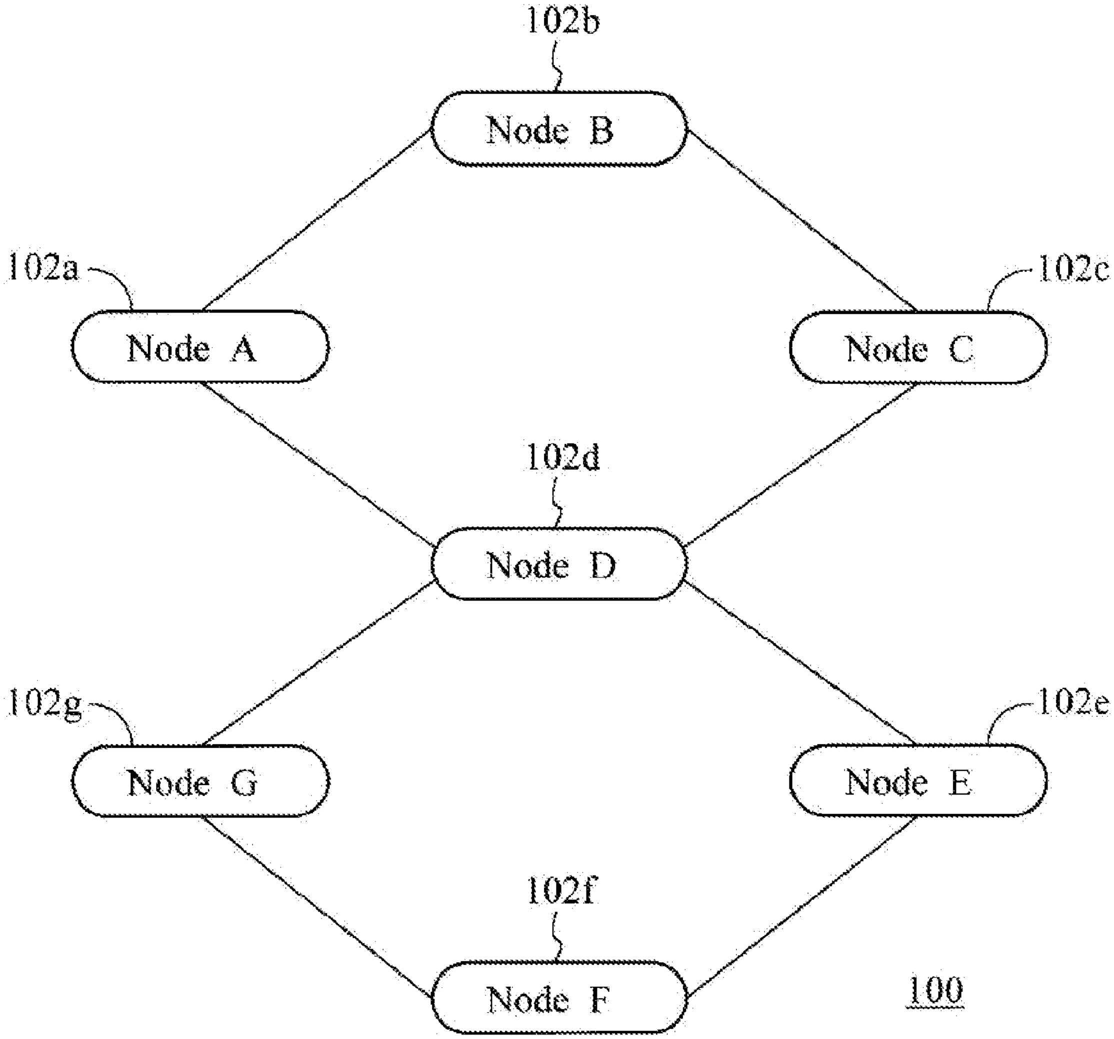


FIG. 1
(PRIOR ART)

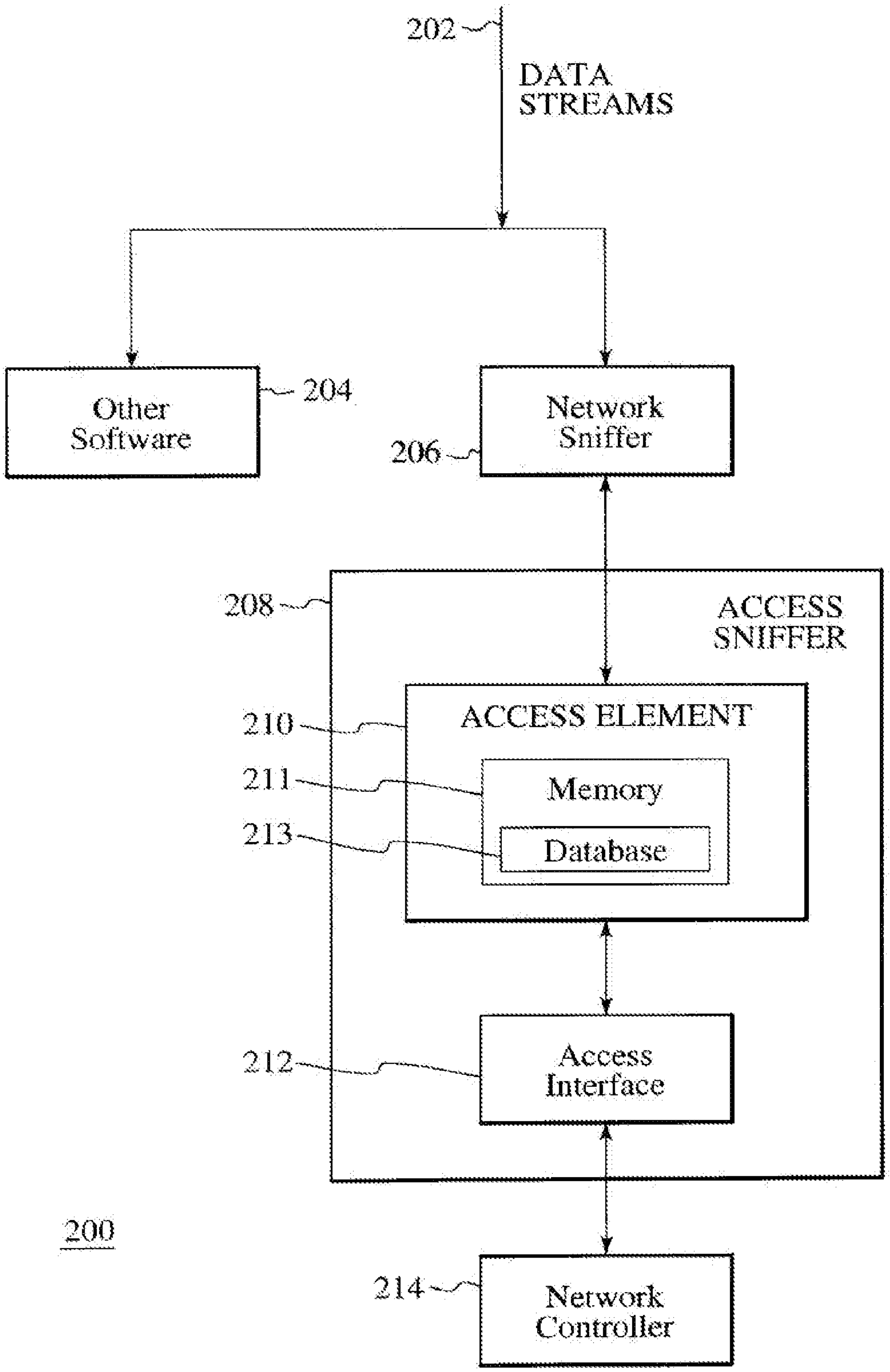

202
DATA STREAMS
Other Software
204
Network Sniffer
206
208
ACCESS SNIFFER
210
ACCESS ELEMENT
211
Memory
213
Database
212
Access Interface
200
214
Network Controller

FIG. 2

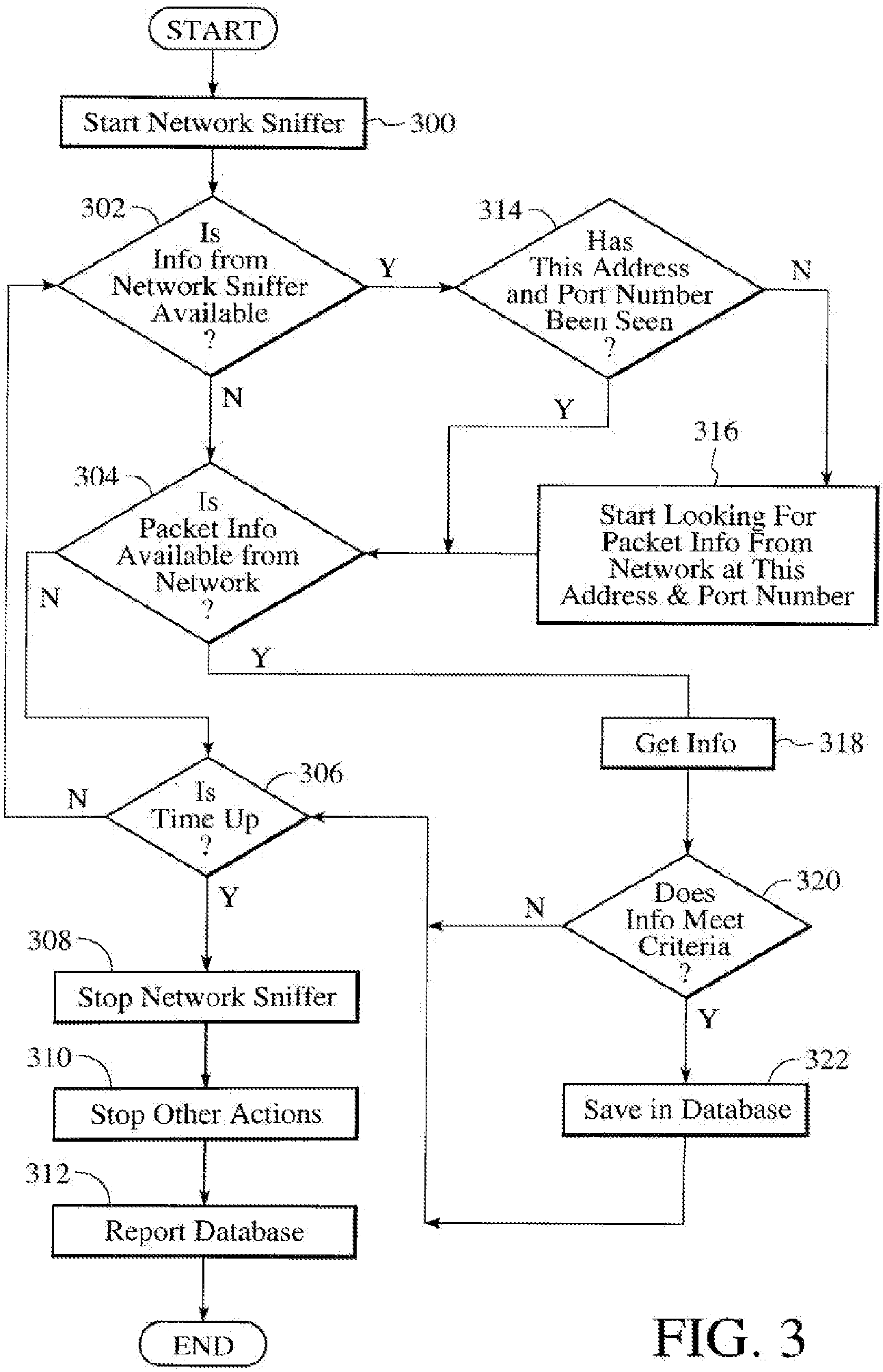

START
Start Network Sniffer
300
302
Is Info from Network Sniffer Available ?
Y
N
314
Has This Address and Port Number Been Seen ?
N
Y
316
304
Is Packet Info Available from Network ?
Start Looking For Packet Info From Network at This Address & Port Number
N
Y
Get Info
318
N
Is Time Up ?
306
Y
320
N
Does Info Meet Criteria ?
Y
308
Stop Network Sniffer
310
Stop Other Actions
322
Save in Database
312
Report Database
END

FIG. 3

NETWORK SNIFFER FOR MONITORING AND REPORTING NETWORK INFORMATION THAT IS NOT PRIVILEGED BEYOND A USER'S PRIVILEGE LEVEL

TECHNICAL FIELD

The present invention is related to locating information in a network environment, particularly available information in a network environment being located by a node in the network through the use of a network sniffer along with an access sniffer.

BACKGROUND

A network environment typically includes multiple nodes in which a node itself can also be a network. Information can be transmitted from one of these nodes which can be received by another node in the network.

Multimedia data streams can be sent through public networks for reception by the general public. The data streams can consist of audio, video, whiteboard, or any other type of digital data. A user on the network can receive these data streams using the appropriate software and with knowledge of the multicast address and port number for the stream. The problem is that it is difficult to locate the desired information since the user must typically know the address and port number for a desired data stream.

A network sniffer is a system and method that is normally used to monitor network activity when resolving network problems or when improving network efficiency. Although the network sniffer can typically access information regarding all data being transferred into the node, access to the network sniffer is generally privileged due to the sensitivity of some of the data on the networks. Thus, there is a need to facilitate ease of access to available information in a network environment. The present invention addresses such a need.

SUMMARY

The present invention is a system and method for locating requested data streams by accessing data monitored by a network sniffer for a particular node and evaluating data which is within the privilege level of a particular user.

A network sniffer is a system and method that is normally used to monitor network activity when resolving network problems or when improving network efficiency. Because of the sensitivity of some of the data on networks, average users are not generally allowed to access network sniffers.

A method according to the present invention for locating available information in a network environment by a node in the network comprising the steps of accessing a network sniffer which can monitor data sent to a node; determining if the data being monitored by the sniffer is privileged; and accessing data which is not privileged beyond the user's level.

The present invention can locate an address and port number of a data packet through the use of the network sniffer, determine if it has already been processed, evaluate the data packet associated with the address and port number, and determine if it meets predetermined criteria. If it does meet the predetermined criteria, then the data is saved in a database.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram representing a network.

FIG. 2 is a functional block diagram of a system according to the present invention.

FIG. 3 is a flow diagram of a method according to the present invention.

DETAILED DESCRIPTION

The illustrative embodiment is related to a system and method for locating available information in a network. The following description is presented to enable on of ordinary skill in the art to make and use the illustrative embodiment and is provided in the context of a patent application and its requirements. Various modifications to the illustrative embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. Thus, the illustrative embodiment is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

FIG. 1 is a diagram depicting a network environment **100**. The network **100** includes multiple nodes **102***a*–**102***d*, in which a node itself can also be a network, as shown by **102***d*–**102***g*. Information, such as video and audio, can be transmitted from one of the nodes such as the node **102***a*. This information can be received by another node in the network such as node **102***f* by using appropriate software and the knowledge of the multicast address and port number for the data stream.

The present invention uses services available through a network sniffer to examine the network connections and list the addresses and port numbers of available data streams such as multicast streams.

The present invention allows non-privileged users to access a network sniffer for gathering public information. It can also allow users to access a network sniffer for gathering information which is available up to the particular privilege level of the user. Public information and information which is available up to the particular privilege level of the user is herein collectively referred to as "available information." Non-public and sensitive information beyond the privilege level of the user is not reported. The available information may be further filtered so that only data meeting predetermined criteria, such as data for particular applications, are reported.

There are many ways in which a piece of information may be deemed privileged. Privileged information can include, but is not limited to, the following:

(1) a packet of information form the network having a destination address that is different from the address of the node the user is using;

(2) a packet of information having a source or destination port number that is reserved for privileged processes;

(3) a packet of information having a source or destination port number that is well known for passing privileged information;

(4) a packet of information which information is scrambled or encrypted;

(5) a packet of information which is not scrambled or encrypted, but contains a recognized structure that indicates that the data is privileged;

(6) a packet of information which is not scrambled or encrypted, but contains a recognized structure that indicates that the destination is some other user.

There are also many ways in which a piece of information may be deemed not privileged or not beyond the particular user's privilege level. Information which is not privileged can include, but is not limited to, the following:

(1) a packet of information which is not encrypted and contains a recognized structure that indicates that the data is not privileged;

(2) a packet of information having a source or destination port number that is well known for passing non-privileged information.

In addition to including the information which is not privileged, information which is not beyond the particular user's privilege level can also include, but is not limited to, the following:

(1) a packet of information which is scrambled or encrypted and can be unscrambled or decrypted with information (a key) provided by the user.

(2) a packet of information which is not scrambled or encrypted, but contains a recognized structure that indicates that the destination is the particular user.

Although the examples described herein are for multicast data, the method and system of the present invention also applies to broadcast, unicast, and other data that can be identified as available to the user.

FIG. 2 is a functional block diagram of a system according to the present invention. The basic components of system **200**, which is preferably located within a node in a network, include a network sniffer **206**, an access sniffer **208**, and a network controller **214**. The access sniffer **208** includes access element **210** and access interface **212**. The access element **210** preferably includes a memory **211** and a database **213**.

Data streams **202** enter the node and are utilized by software **204** in the system **200**. Copies of the data streams **202**, or portions thereof, can be accessed by the network sniffer **206**. The access sniffer **208** can then access the data streams being evaluated by the network sniffer **206**. The access element **210** accesses the network sniffer **206** and filters out unavailable information by using information such as address and port numbers gathered by the network sniffer **206**. Unavailable information includes information which is non-public or beyond the privilege level of the particular user.

The access element **210** can evaluate data streams which are public information to determine if the data streams meet a predetermined criteria, such as multicast real time protocol (RTP), for video and audio. Examples of addresses which the access elements **210** can utilize include multicast and unicast addresses.

Access element **210** preferably removes all unacceptable data streams. It preferably executes virtually all the major functions of the access sniffer **208**. The access element **210** can exclude data streams which are not targeted for this particular node, and exclude addresses destined for other software on the node. It can also execute unicast addresses if one of the predetermined criteria is to look for a multicast addresses. It can run tests on the data streams to determine if the data streams are valid. If it is not valid, then the access element **210** can exclude it. If it is valid, then it can determine whether the address and port number associated with that valid data stream are information available to the particular user. Access element **210** can also monitor the port number for data streams which meet the predetermined criteria.

The access element **210** can also check to see if a particular address and port number have been evaluated in the past. If so, then it ignores the previously evaluated address and port number. If the address and port number have not been previously evaluated, and they also meet the predetermined criteria, then these data streams are kept in the data base **213**. Examples of predetermined criteria include a particular privilege level such as public information, a type of information such as a particular node name, type name, or user name of the particular source.

Once access element **210** finds data streams which meet the predetermined criteria, it can then save that information along with the port number and the address associated with it. When the access element **210** has completed its functions, it preferably saves the filtered data and transfers only the information available to the particular user to the access interface **212**. It is preferable that the access element **210** does not transfer encrypted information or password information to the access interface **212**.

The access element **210** can time itself for a limited amount of time for execution. Once the predetermined time period has expired, the access element **210** is complete, and it can save and transfer the appropriate information to the access interface **212**.

One example of how the access element **210** maintains information in the database **213** is to open what is referred to in UNIX as a socket. The access element **210** can obtain a body of a data packet from the socket by looking for a specific port number and address. If the data stream located in the data base **213** is a particular type of data stream, such as a type name, a particular node name, or user name of the particular source, then that type can also be added to the data base **213**.

At the end of the predetermined time, the access element **210** preferably stops the process, closes all the sockets it opened, evaluates the database **213**, converts the requested information to a string, and passes it to the access interface **212**.

The information in the network sniffer **206** and the access element **210** are preferably privileged information, thus the user preferably does not have direct access to the access element **210**, but rather utilizes the access interface **212** to access only the information which is either public or within that particular user's privilege level. The information within the access interface **212** can then be transferred to the network controller **214**. Although the access interface is shown in FIG. 2 to be included in the access sniffer **208**, the access interface **212** can be a separate component and the access element **210** can then function as the access sniffer **208**.

FIG. 3 is a flow diagram of a method according to the present invention. The method shown in FIG. 3 is merely one example of obtaining the desired results according to the present invention. In this example, the network sniffer is started via step **300**. It is then determined if the information from the network sniffer is available via step **302**. The access element **210** of FIG. 2 would preferably perform this determination. As previously stated, available information means either public information or information which is within the particular user's privilege level. If the information from the network sniffer is not available, then it is determined, preferably by access element **210** of FIG. 2, if the packet information is available form the network via step **304**. Note that the order of step **304** and **302** is completely interchangeable without affecting the results.

If the information form the data packet is not available, then it is determined if the predetermined time has elapsed via step **306**. If the predetermined time has not elapsed, then another data packet is evaluated by the access element **210** from the network sniffer via step **302** and **304**. If the predetermined time has elapsed, then the network sniffer is stopped via step **308** and any other actions which are in progress are also stopped via step **310**. The other actions include actions via steps **314-322** which would also preferably be performed by access element **210** of FIG. **2**. Information in the database is then reported to the access interface **212** of FIG. **2** via step **312**.

If information from the network sniffer is available via step **302**, then it is determined whether the address and port number of the data stream have previously been evaluated via step **314**. If it has been previously evaluated, then it is determined if the packet information is available from the network via step **304**. If it has not been previously evaluated, then the packet information from the network is looked for at the particular address and port number via step **316**. Then it is determined if the packet information is available from the network via step **304**. If the packet information is available from the network to the particular user via step **304**, then the information is retrieved via step **318**. It is then determined if the information meets the predetermined criteria via step **320**. If it does not, then it is determined whether the predetermined time has elapsed via step **306**. If, however, the information does meet the predetermined criteria, then it is saved in the data base via step **322**. Again, it is determined if the predetermined time has elapsed via step **306**. If so, then steps **308**–**312** are executed. If the predetermined time has not elapsed, then the next piece of information from the network sniffer is evaluated via step **302**.

Although the system and method has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations to the embodiments and those variations would be within the spirit and scope of the above-described system and method. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.

What is claimed is:

1. A method for locating available information in a network environment by a user at a node in the network, the method comprising the steps of:

a) accessing the network sniffer that is monitoring data sent to the node;

b) determining if the data being monitored by the network sniffer is privileged; and

c) accessing data which is not privileged beyond the user's privileged level.

2. A method for locating at a node requested data from among data sent to the node in a network environment, the requested data being located for providing access thereto by a user at the node, the method comprising the steps of:

a) accessing a network sniffer that is monitoring the data sent to the node;

b) determining if the data being monitored by the network sniffer is privileged or public;

c) determining if the data being monitored includes the available information if the data is privileged, wherein a privilege level of the available information is at or below a privilege level of the user;

d) locating addresses and port numbers of the available information, wherein the available information from among the data being monitored includes the data which is public;

e) reporting the available information including the corresponding addresses and port numbers; and

f) providing access to the available information, wherein the available information includes the requested data and wherein the requested data is not privileged beyond the user's privilege level.

3. The method of claim **2**, wherein the providing access step (f) further includes the step (f1) of transferring the requested data including the corresponding addresses and port numbers to the user at the node.

4. The method of claim **3** wherein the data determining step (b) further includes step (b2) of determining if a particular address and port number have already been processed.

5. The method of claim **3**, wherein the data determining step (b) further includes step (b2) of evaluating a data packet associated with the address and port number.

6. The method of claim **2**, wherein the reporting step (e) further includes the step (e1) of determining if the requested data which meets a predetermined criteria is included in the available information by determining if the data being monitored which comprises the available information meets the predetermined criteria.

7. The method of claim **6**, wherein the reporting step (e) further includes the step (e2) of storing the requested data in a database if the data being monitored, which comprises the available information, meets the predetermined criteria.

8. The method of claim **6**, wherein the predetermined criteria includes a data valid requirement.

9. The method of claim **6**, wherein the requested data includes data which is audio data.

10. The method of claim **6**, wherein the requested data includes data which is video data.

11. The method of claim **2**, further comprising the step (g) of determining if time is up for stopping the accessing of the sniffer in step (a) after a predetermined time period and, if so, terminating all actions (steps a–f) including accessing the sniffer.

12. A system for locating available information in the network environment by a user in a node, the system comprising:

means for monitoring data which is sent to the node; and

means coupled to the monitoring means for accessing the data, wherein the accessing means determines if the data being monitored by the monitoring means is privileged beyond the user's privilege level, and wherein the accessing means reports the data which is not privileged beyond the user's privilege level and, with the data, the accessing means reports addresses and port numbers corresponding thereto.

13. The system of claim **12**, further comprising a memory means coupled to the accessing means for storing the data being accessed if the accessed data meets a predetermined criteria.

14. The system of claim **12**, further comprising an interface means for transferring the data which is not privileged beyond the user's privilege level.

15. A system for locating available information in the network environment by user data node, the system comprising:

a first sniffer for monitoring data which is sent to a node; and

a second sniffer coupled to the first sniffer for accessing the data which is being monitored by the first sniffer, wherein the second sniffer determines if the data being monitored is privileged beyond the user's privilege level, and wherein the accessing means reports the data which is not privileged beyond the user's privilege level and, with the data, the accessing means reports addresses and port numbers corresponding thereto.

16. The system of claim **15**, wherein the second sniffer further comprises a memory for storing accessed data if the accessed data meets a predetermined criteria.

17. The system of claim **15**, further comprising an interface for transferring the data which is not privileged beyond the user's privilege level.

18. A system for locating available information in a network environment by a user edit node, the system comprising:

accessing means for accessing data which is being monitored by a monitoring means, wherein the accessing means determines is the data being monitored is privileged beyond the user's privilege level, and wherein the accessing means reports the data which is not privileged beyond the user's privileged level and, with the data, the accessing means reports addresses and port numbers corresponding thereto; and

interface means coupled to the accessing means for transferring accessed data which is not beyond the user's privilege level.

19. A computer readable medium containing program instructions for locating at a node requesting data from among data sent to the node in and network environment, the requested data being located for providing access thereto by a user at a node, the program instructions for:

a) accessing the networks never that is monitoring the data sent to the node;

b) determining if the data being monitored by the network sniffers is privileged or public;

c) determining if it data being monitored includes the available information if the data is privileged, wherein a privilege level of the available information is at or below a privilege level of the user;

d) locating addresses and port numbers of the available information wherein the available information from among the data being monitored includes the data which is public;

e) reporting the available information including the corresponding addresses and port numbers; and

f) providing access to the available information, wherein the available information includes the requested data and wherein the requested data is not privileged beyond the user's privilege level.

* * * * *

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L 29/06, 29/08	A1	(11) International Publication Number: WO 98/34385 (43) International Publication Date: 6 August 1998 (06.08.98)

(21) International Application Number: PCT/US98/01797

(22) International Filing Date: 30 January 1998 (30.01.98)

(30) Priority Data:

60/036,661	30 January 1997 (30.01.97)	US
60/036,662	30 January 1997 (30.01.97)	US
08/818,769	14 March 1997 (14.03.97)	US

(71) Applicant: VXTREME, INC. [US/US]; 675 Almanor Avenue, Sunnyvale, CA 94086 (US).

(72) Inventors: VELLANKI, Srinivas, Prasad; 400 Glenmoor Circle, Milpitas, CA 95035 (US). CANNON, Anthony, William; Apartment 334, 1600 Villa Street, Mountain View, CA 94041 (US). RAVI, Hemanth, Srinivas; 444 Glenmoor Circle, Milpitas, CA 95035 (US). KLEMETS, Anders, Edgar; 655 S. Fair Oaks Avenue, #K-313, Sunnyvale, CA 94086 (US).

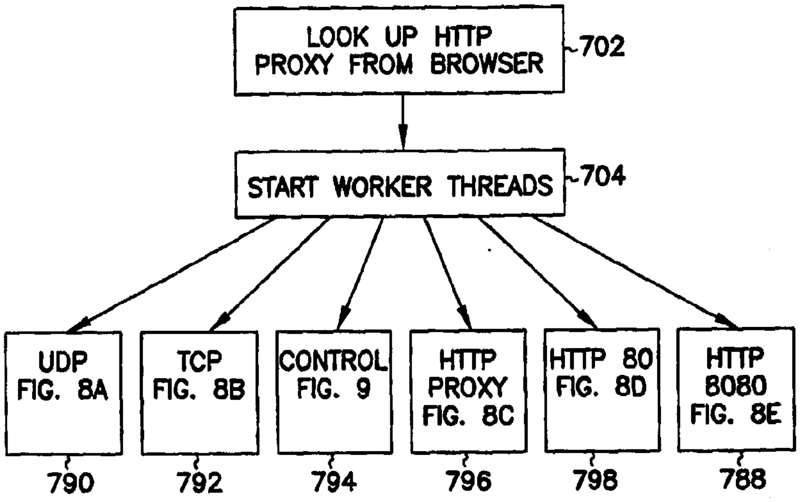
(74) Agent: VIKSNINS, Ann, S.; Schwegman, Lundberg, Woessner & Kluth, P.O. Box 2938, Minneapolis, MN 55402 (US).

(81) Designated States: JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published

With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: AUTOMATIC DETECTION OF PROTOCOLS IN A NETWORK



(57) Abstract

A method in a computer network for automatically detecting a most advantageous protocol for communication by a client computer, said client computer being configured to be coupled to a server computer via a computer network. The method includes initiating a plurality of protocol threads for sending from the client computer to the server computer, a plurality of data requests. Each of the data requests employs a different protocol and a different connection. The data requests are configured to solicit, responsive to the data requests, a set of responses from the server computer. Each of the responses employs a protocol associated with a respective one of the data requests. The method further includes receiving at the client computer at least a subset of the responses. The method also includes initiating a control thread at the client computer. The control thread monitors the subset of the responses as each response is received from the server computer to select the most advantageous protocol from protocols associated with the subset of the responses, wherein the most advantageous protocol is determined based on a predefined protocol priority.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania
AM	Armenia
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CU	Cuba
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GH	Ghana
GN	Guinea
GR	Greecc
HU	Hungary
IE	Ireland
IL	Israel
IS	Iceland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakstan
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
MK	The former Yugoslav Republic of Macedonia
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TM	Turkmenistan
TR	Turkey
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam
YU	Yugoslavia
ZW	Zimbabwe

AUTOMATIC DETECTION OF PROTOCOLS IN A NETWORK

BACKGROUND OF THE INVENTION

The present invention relates to data communication in a computer network. More particularly, the present invention relates to improved methods and apparatus for permitting a client computer in a client-server architecture computer network to automatically detect the most advantageous protocol, among the protocols available, for use in communicating with the server irrespective whether there exist firewalls or proxies in the network.

Client-server architectures are well known to those skilled in the computer art. For example, in a typical computer network, one or more client computers may be coupled to any number of server computers. Client computers typically refer to terminals or personal computers through which end users interact with the network. Server computers typically represent nodes in the computer network where data, application programs, and the like reside. Server computers may also represent nodes in the network for forwarding data, programs, and the like from other servers to the requesting client computers.

To facilitate discussion, Fig. 1 illustrates a computer network 100, representing for example a subset of an international computer network popularly known as the Internet. As is well known, the Internet represents a well-known international computer network that links, among others, various military, governmental, educational, nonprofit, industrial and financial institutions, commercial enterprises, and individuals. There are shown in Fig. 1 a server 102, a server 104, and a client computer 106. Server computer 104 is separated from client computer 106 by a firewall 108, which may be implemented in either software or hardware, and may reside on a computer and/or circuit between client computer 106 and server computer 104.

Firewall 108 may be specified, as is well known to those skilled in the art, to prevent certain types of data and/or protocols from traversing through it. The specific data and/or protocols prohibited or permitted to traverse firewall

108 depend on the firewall parameters, which are typically set by a system administrator responsible for the maintenance and security of client computer 106 and/or other computers connected to it, e.g., other computers in a local area network. By way of example, firewall 108 may be set up to prevent TCP, UDP, or HTTP (Transmission Control Protocol, User Datagram Protocol, and Hypertext Transfer Protocol, respectively) data and/or other protocols from being transmitted between client computer 106 and server 104. The firewalls could be configured to allow specific TCP or UDP sessions, for example outgoing TCP connection to certain ports, UDP sessions to certain ports, and the like.

Without a firewall, any type of data and/or protocol may be communicated between a client computer and a server computer if appropriate software and/or hardware are employed. For example, server 102 resides on the same side of firewall 108 as client computer 106, i.e., firewall 108 is not disposed in between the communication path between server 102 and client computer 106. Accordingly, few if any, of the protocols that client computer 106 may employ to communicate with server 102 may be blocked.

As is well known to those skilled in the art, some computer networks may be provided with proxies, i.e., software codes or hardware circuities that facilitate the indirect communication between a client computer and a server around a firewall. With reference to Fig. 1, for example, client computer 106 may communicate with server 104 through proxy 120. Through proxy 120, HTTP data, which may otherwise be blocked by firewall 108 for the purpose of this example, may be transmitted between client computer 106 and server computer 104.

In some computer networks, one or more protocols may be available for communication between the client computer and the server computer. For certain applications, one of these protocols, however, is often more advantageous, i.e., suitable, than others. By way of example, in applications involving real-time data rendering (such as rendering audio, video, and/or annotation data as they are streamed from a server), it is highly preferable that the client computer executing that application selects a protocol that permits the greatest degree of control over the transmission of data packets and/or enables data transmission to occur at the highest possible rate. This is because these applications are fairly demanding in terms of their bit rate and connection reliability requirements. Accordingly, the quality of the data rendered, e.g., the video and/or audio clips played, often depends on whether the user has successfully configured the client computer to receive data from the server computer using the most advantageous protocol available.

In the prior art, the selection of the most advantageous protocol for communication between client computer 106 and server computer 104 typically requires a high degree of technical sophistication on the part of the user of client computer 106. By way of example, it is typically necessary in the prior art for the user of client computer 106 to understand the topology of computer network 100, the protocols available for use with the network, and/or the protocols that can traverse firewall 108 before that user can be expected to configure his client computer 106 for communication.

This level of technical sophistication is, however, likely to be beyond that typically possessed by an average user of client computer 106. Accordingly, users in the prior art often find it difficult to configure their client computers even for simple communication tasks with the network. The difficulties may be encountered for example during the initial setup or whenever there are changes in the topology of computer network 100 and/or in the technology employed to transmit data between client computer 106 and server 104. Typically, expert and expensive assistance is required, if such assistance is available at all in the geographic area of the user.

Furthermore, even if the user can configure client computer 106 to communicate with server 104 through firewall 108 and/or proxy 120, there is no assurance that the user of client computer 106 has properly selected, among the protocols available, the most advantageous protocol communication (e.g., in terms of data transmission rate, transmission control, and the like). As mentioned earlier, the ability to employ the most advantageous protocol for communication, while desirable for most networking applications, is particularly critical in applications such as real-time data rendering (e.g., rendering of audio, video, and/or annotation data as they are received from the server). If a less than optimal protocol is chosen for communication, the quality of the rendered data, e.g., the video clips and/or audio clips, may suffer.

In view of the foregoing, there are desired improved techniques for permitting a client computer in a client-server network to efficiently, automatically, and appropriately select the most advantageous protocol to communicate with a server computer.

SUMMARY OF THE INVENTION

The invention relates, in one embodiment, to a method in a computer network for automatically detecting a most advantageous protocol for communication by a client computer. The client computer is configured to be coupled to a server computer via a computer network. The method includes sending from the client computer to the server computer, a plurality of data requests. Each of the data requests employs a different protocol and a different connection. The data requests are configured to solicit, responsive to the data requests, a set of responses from the server computer. Each of the responses employs a protocol associated with a respective one of the data requests.

The method further includes receiving at the client computer at least a subset of the responses. The method also includes monitoring the subset of the responses as each response is received from the server computer to select the most advantageous protocol from protocols associated with the subset of the responses, wherein the most advantageous protocol is determined based on a predefined protocol priority.

In another embodiment, the invention relates to a method in a computer network for automatically detecting a most advantageous protocol for communication by a client computer. The client computer is configured to be coupled to a server computer via a computer network. The method includes sending from the client computer to the server computer, a plurality of data requests. Each of the data requests employs a different protocol and a different connection.

The method further includes receiving at least a subset of the data requests at the server computer. The method additionally includes sending a set of responses form the server computer to the client computer. The set of responses is responsive to the subset of the data requests. Each of the responses employs a protocol associated with a respective one of the subset of the data requests. The method also includes receiving at the client computer at least a subset of the responses. There is further included selecting, for the communication between the client computer and the server computer, the most advantageous protocol from protocols associated with the subset of the responses, wherein the most advantageous protocol is determined based on a predefined protocol priority.

In yet another embodiment, the invention relates to a computer-readable medium containing computer-readable instructions for automatically detecting a most advantageous protocol for communication by a client computer. The client computer is configured to be coupled to a server computer via a computer network. The computer-readable instructions comprise computer-readable instructions for sending in a substantially parallel manner, from the client computer to the server computer, a plurality of data requests. Each of the data requests employs a different protocol and a different connection. The data requests are configured to solicit, responsive to the data requests, a set of responses from the server computer. Each of the responses employs a protocol associated with a respective one of the data requests.

The computer-readable medium further includes computer-readable instructions for receiving at the client computer at least a subset of the responses. There is further included computer-readable instructions for monitoring the subset of the responses as each response is received from the server computer to select the most advantageous protocol from protocols associated with the subset of the responses, wherein the most advantageous protocol is determined based on a predefined protocol priority.

These and other features of the present invention will be described in more detail below in the detailed description of the invention and in conjunction with the following figures.

BRIEF DESCRIPTION OF THE DRAWINGS

To facilitate discussion, Fig. 1 illustrates a computer network, representing for example a portion of an international computer network popularly known as the Internet.

Fig. 2 is a block diagram of an exemplar computer system for carrying out the autodetect technique according to one embodiment of the invention.

Fig. 3 illustrates, in accordance with one embodiment, the control and data connections between a client application and a server computer when no firewall is provided in the network.

Fig. 4 illustrates another network arrangement wherein control and data connections are established through a firewall.

Figs. 5A-B illustrate another network arrangement wherein media control commands and media data may be communicated between a client computer and server computer using the HTTP protocol.

Figs. 5C-D illustrate another network arrangement wherein multiple HTTP control and data connections are multiplexed through a single HTTP port.

Fig. 6 illustrates another network arrangement wherein control and data connections are transmitted between the client application and the server computer via a proxy.

Fig. 7 depicts, in accordance with one embodiment of the present invention, a simplified flowchart illustrating the steps of the inventive autodetect technique.

Fig. 8A depicts, in accordance with one aspect of the present invention, the steps involved in executing the UDP protocol thread of Fig. 7.

Fig. 8B depicts, in accordance with one aspect of the present invention, the steps involved in executing the TCP protocol thread of Fig. 7.

Fig. 8C depicts, in accordance with one aspect of the present invention, the steps involved in executing the HTTP protocol thread of Fig. 7.

Fig. 8D depicts, in accordance with one aspect of the present invention, the steps involved in executing the HTTP 80 protocol thread of Fig. 7.

Fig. 8E depicts, in accordance with one aspect of the present invention, the steps involved in executing the HTTP 8080 protocol thread of Fig. 7.

Fig. 9 illustrates, in accordance with one embodiment of the present invention, the steps involved in executing the control thread of Fig. 7.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention will now be described in detail with reference to a few preferred embodiments thereof as illustrated in the accompanying drawings. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without some or all of these specific details. In other instances, well-known process steps have not been described in detail in order to not unnecessarily obscure the present invention.

In accordance with one aspect of the present invention, the client computer in a heterogeneous client-server computer network (e.g., client computer 106 in Fig. 1) is provided with an autodetect mechanism. When executed, the autodetect mechanism advantageously permits client computer 106 to select, in an efficient and automatic manner, the most advantageous protocol for communication between the client computer and its server. Once the most advantageous protocol is selected, parameters pertaining to the selected protocol are saved to enable the client computer, in future sessions, to employ the same selected protocol for communication.

In accordance with one particular advantageous embodiment, the inventive autodetect mechanism simultaneously employs multiple threads, through multiple connections, to initiate communication with the server computer, e.g., server 104. Each thread preferably employs a different protocol and requests the server computer to respond to the client computer using the protocol associated with that thread. For example, client computer 106 may, employing the autodetect mechanism, initiate five different threads, using respectively the TCP, UDP, one of HTTP and HTTP proxy, HTTP through port (multiplex) 80, and HTTP through port (multiplex) 8080 protocols to request server 104 to respond.

Upon receiving a request, server 104 responds with data using the same protocol as that associated with the thread on which the request arrives. If one or more protocols is blocked and fails to reach server 104 (e.g., by a firewall), no response employing the blocked protocol would of course be transmitted from server 104 to client computer 106. Further, some of the protocols transmitted from server 104 to client computer 106 may be blocked as well. Accordingly, client computer may receive only a subset of the responses sent from server 104.

In one embodiment, client computer 106 monitors the set of received responses. If the predefined "best" protocol is received, that protocol is then selected for communication by client computer 106. The predefined "best" protocol may be defined in advance by the user and/or the application program. If the predefined "best" protocol is, however, blocked (as the request is transmitted from the client computer or as the response is transmitted from the server, for example) the most advantageous protocol may simply be selected from the set of protocols received back by the client computer. In one embodiment, the selection may be made among the set of protocols received back by the client computer within a predefined time period after the requests are sent out in parallel.

The selection of the most advantageous protocol for communication among the protocols received by client computer 106 may be performed in accordance with some predefined priority. For example, in the real-time data rendering application, the UDP protocol may be preferred over TCP protocol, which may in turn be preferred over the HTTP protocol. This is because the UDP protocol typically can handle a greater data transmission rate and may allow client computer 106 to exercise a greater degree of control over the transmission of data packets.

HTTP data, while popular nowadays for use in transmitting web pages, typically involves a higher number of overhead bits, making it less efficient relative to the UDP protocol for transmitting real-time data. As is known, the HTTP protocol is typically built on top of TCP. The underlaying TCP protocol typically handles the transmission and retransmission requests of individual data packets automatically. Accordingly, the HTTP protocol tends to reduce the degree of control client computer 106 has over the transmission of the data packets between server 104 and client computer 106. Of course other priority schemes may exist for different applications, or even for different real-time data rendering applications.

In one embodiment, as client computer 106 is installed and initiated for communication with server 104 for the first time, the autodetect mechanism is invoked to allow client computer 106 to send transmission requests in parallel (e.g., using different protocols over different connections) in the manner discussed earlier. After server 104 responds with data via multiple connections/protocols and the most advantageous protocol has been selected by client computer 106 for communication (in accordance with some predefined priority), the parameters associated with the selected protocol are then saved for future communication.

Once the most advantageous protocol is selected, the autodetect mechanism may be disabled, and future communication between client computer 106 and server 104 may proceed using the selected most advantageous protocol without further invocation of the autodetect mechanism. If the topology of computer network 100 changes and communication using the previously selected "most advantageous" protocol is no longer appropriate, the autodetect mechanism may be executed again to allow client computer 106 to ascertain a new "most advantageous" protocol for communication with server 104. In one embodiment, the user of client computer 106 may, if desired, initiate the autodetect mechanism at any time in order to enable client computer 106 to update the "most advantageous" protocol for communication with server 104 (e.g., when the user of client computer 106 has reasons to suspect that the previously selected "most advantageous" protocol is no longer the most optimal protocol for communication).

The inventive autodetect mechanism may be implemented either in software or hardware, e.g., via an IC chip. If implemented in software, it may be carried out by any number of computers capable of functioning as a client computer in a computer network. Fig. 2 is a block diagram of an exemplar computer system 200 for carrying out the autodetect technique according to one embodiment of the invention. Computer system 200, or an analogous one, may be employed to implement either a client or a server of a computer network. The computer system 200 includes a digital computer 202, a display screen (or monitor) 204, a printer 206, a floppy disk drive 208, a hard disk drive 210, a network interface 212, and a keyboard 214. The digital computer 202 includes a microprocessor 216, a memory bus 218, random access memory (RAM) 220, read only memory (ROM) 222, a peripheral bus 224, and a keyboard controller 226. The digital computer 200 can be a personal computer (such as an Apple computer, e.g., an Apple Macintosh, an IBM personal computer, or one of the compatibles thereof), a workstation computer (such as a Sun Microsystems or Hewlett-Packard workstation), or some other type of computer.

The microprocessor 216 is a general purpose digital processor which controls the operation of the computer system 200. The microprocessor 216 can be a single-chip processor or can be implemented with multiple components. Using instructions retrieved from memory, the microprocessor 216 controls the reception and manipulation of input data and the output and display of data on output devices.

The memory bus 218 is used by the microprocessor 216 to access the RAM 220 and the ROM 222. The RAM 220 is used by the microprocessor 216 as a general storage area and as scratch-pad memory, and can also be used to store input data and processed data. The ROM 222 can be used to store instructions or program code followed by the microprocessor 216 as well as other data.

The peripheral bus 224 is used to access the input, output, and storage devices used by the digital computer 202. In the described embodiment, these devices include the display screen 204, the printer device 206, the floppy disk drive 208, the hard disk drive 210, and the network interface 212, which is employed to connect computer 200 to the network. The keyboard controller 226 is used to receive input from keyboard 214 and send decoded symbols for each pressed key to microprocessor 216 over bus 228.

The display screen 204 is an output device that displays images of data provided by the microprocessor 216 via the peripheral bus 224 or provided by other components in the computer system 200. The printer device 206 when operating as a printer provides an image on a sheet of paper or a similar surface. Other output devices such as a plotter, typesetter, etc. can be used in place of, or in addition to, the printer device 206.

The floppy disk drive 208 and the hard disk drive 210 can be used to store various types of data. The floppy disk drive 208 facilitates transporting such data to other computer systems, and hard disk drive 210 permits fast access to large amounts of stored data.

The microprocessor 216 together with an operating system operate to execute computer code and produce and use data. The computer code and data may reside on the RAM 220, the ROM 222, the hard disk drive 220, or even on another computer on the network. The computer code and data could also reside on a removable program medium and loaded or installed onto the computer system 200 when needed. Removable program mediums include, for example, CD-ROM, PC-CARD, floppy disk and magnetic tape.

The network interface circuit 212 is used to send and receive data over a network connected to other computer systems. An interface card or similar device and appropriate software implemented by the microprocessor 216 can be used to connect the computer system 200 to an existing network and transfer data according to standard protocols.

The keyboard 214 is used by a user to input commands and other instructions to the computer system 200. Other types of user input devices can also be used in conjunction with the present invention. For example, pointing devices such as a computer mouse, a track ball, a stylus, or a tablet can be used to manipulate a pointer on a screen of a general-purpose computer.

The invention can also be embodied as computer-readable code on a computer-readable medium. The computer-readable medium is any data storage device that can store data which can thereafter be read by a computer system. Examples of the computer-readable medium include read-only memory, random-access memory, CD-ROMs, magnetic tape, optical data storage devices. The computer-readable code can also be distributed over a network coupled computer system so that the computer-readable code is stored and executed in a distributed fashion.

Figs. 3-6 below illustrate, to facilitate discussion, some possible arrangements for the transmission and receipt of data in a computer network. The arrangements differ depending on which protocol is employed and the configuration of the network itself. Fig. 3 illustrates, in accordance with one embodiment, the control and data connections between a client application 300 and server 302 when no firewall is provided in the network.

Client application 300 may represent, for example, the executable codes for executing a real-time data rendering program such as the Web Theater Client 2.0, available from VXtreme, Inc. of Sunnyvale, California. In the example of Fig. 3, client application 300 includes the inventive autodetect mechanism and may represent a plug-in software module that may be installed onto a browser 306. Browser 306 may represent, for example, the application program which the user of the client computer employs to navigate the network. By way of example, browser 306 may represent one of the popular Internet browser programs, such as Netscape™ by Netscape Communications Inc. of Mountain View, California or Microsoft Explorer by Microsoft Corporation of Redmond, Washington.

When the autodetect mechanism of client application 300 is executed in browser 306 (e.g., during the set up of client application 300), client application 300 sends a control request over control connection 308 to server 302. Although multiple control requests are typically sent in parallel over multiple control connections using different protocols as discussed earlier, only one control request is depicted in Fig. 3 to facilitate ease of illustration.

The protocol employed to send the control request over control connection 308 may represent, for example, TCP, or HTTP. If UDP protocol is requested from the server, the request from the client may be sent via the control connection using for example the TCP protocol. Initially, each control request from client application 300 may include, for example, the server name that identifies server 302, the port through which control connection may be established, and the name of the video stream requested by client application 300. Server 302 then responds with data via data connection 310.

In Fig. 3, it is assumed that no proxies and/or firewalls exist. Accordingly, server 302 responds using the same protocol as that employed in the request. If the request employs TCP, however, server 302 may attempt to respond using either UDP or TCP data connections (depending on the specifics of the request). The response is sent to client application via data connection 310. If the protocol received by the client application is subsequently selected to be the "most advantageous" protocol, subsequent communication between client application 300 and server 302 may take place via control connection 308 and data connection 310. Subsequent control requests sent by client application 300 via control connection 308 may include, for example, stop, play, fast forward, rewind, pause, unpause, and the like. These control requests may be utilized by server 302 to control the delivery of the data stream from server 302 to client application 300 via data connection 310.

It should be noted that although only one control connection and one data connection is shown in Fig. 3 to simplify illustration, multiple control and data connections utilizing the same protocol may exist during a data rendering session. Multiple control and data connections may be required to handle the multiple data streams (e.g., audio, video, annotation) that may be needed in a particular data rendering session. If desired, multiple client applications 300 may be installed within browser 306, e.g., to simultaneously render multiple video clips, each with its own sound and annotations.

Fig. 4 illustrates another network arrangement wherein control and data connections are established through a firewall. As mentioned earlier, a firewall may have policies that restrict or prohibit the traversal of certain types of data and/or protocols. In Fig. 4, a firewall 400 is disposed between client application 300 and server 402. Upon execution, client application 300 sends control request using a given protocol via firewall 400 to server 402. Server 402 then responds with data via data connection 410, again via firewall 400.

If the data and/or protocol can be received by the client computer through firewall 400, client application 300 may then receive data from server 402 (through data connection 408) in the same protocol used in the request. As before, if the request employs the TCP protocol, the server may respond with data connections for either TCP or UDP protocol (depending on the specifics of the request). Protocols that may traverse a firewall may include one or more of the following: UDP, TCP, and HTTP.

In accordance with one aspect of the present invention, the HTTP protocol may be employed to send/receive media data (video, audio, annotation, or the like) between the client and the server. Fig. 5A is a prior art drawing illustrating how a client browser may communicate with a web server using a port designated for communication. In Fig. 5, there is shown a web server 550, representing the software module for serving web pages to a browser application 552. Web server 550 may be any of the commercially available web servers that are available from, for example, Netscape Communications Inc. of Mountain View, California or Microsoft Corporation of Redmond, Washington. Browser application 552 represents for example the Netscape browser from the aforementioned Netscape Communications, Inc., or similarly suitable browser applications.

Through browser application 552, the user may, for example, obtain web pages pertaining to a particular entity by sending an HTTP request (e.g., GET) containing the URL (uniform resource locator) that identifies the web page file. The request sent via control connection 553 may arrive at web server 550 through the HTTP port 554. HTTP port 554 may represent any port through which HTTP communication is enabled. HTTP port 554 may also represent the default port for communicating web pages with client browsers. The HTTP default port may represent, for example, either port 80 or port 8080 on web server 550. As is known, one or both of these ports on web server 550 may be available for web page communication even if there are firewalls disposed between the web server 550 and client browser application 552, which otherwise block all HTTP traffic in other ports. Using the furnished URL, web server 550 may then obtain the desired web page(s) for sending to client browser application 552 via data connection 556.

The invention, in one embodiment, involves employing the HTTP protocol to communicate media commands from a browser application or browser plug-in to the server. Media commands are, for example, PLAY, STOP, REWIND, FAST FORWARD, and PAUSE. The server computer may represent, for example, a web server. The server computer may also represent a video server for streaming video to the client computer. Through the use of the HTTP protocol the client computer may successfully send media control requests and receive media data through any HTTP port. If the default HTTP port, e.g., port 80 or 8080, is specified, the client may successfully send media control requests and receive media data even if there exists a firewall or an HTTP Proxy disposed in between the server computer and the client computer, which otherwise blocks all other traffic that does not use the HTTP protocol. For example, these firewalls or HTTP Proxies do not allow regular TCP or UDP packets to go through.

As is well known to those skilled, the HTTP protocol, as specified by for example the Internet Request For Comments RFC 1945 (T. Berners-Lee et al.), typically defines only three types of requests to be sent from the client computer to the server, namely GET, POST, and HEAD. The POST command, for instance, is specified in RFC 1945 to be composed of a Request-Line, one or more Headers and Entity-Body. To send media commands like PLAY, REWIND, etc., the invention in one embodiment sends the media command as part of the Entity-Body of the HTTP POST command. The media command can be in any format or protocol, and can be, for instance, in the same format as that employed when firewalls are not a concern and plain TCP protocol can be used. This format can be, for example, RTSP (Real Time Streaming Protocol).

When a server gets an HTTP request, it answers the client with an HTTP Response. Responses are typically composed of a Status-Line, one or more headers, and an Entity-Body. In one embodiment of this invention, the response to the media commands is sent as the Entity-Body of the response to the original HTTP request that carried the media command.

Fig. 5B illustrates this use of HTTP for sending arbitrary media commands. In Fig. 5B, the plug-in application 560 within client browser application 562 may attempt to receive media data (e.g., video, audio, annotation, or the like) by first sending an HTTP request to server 564 via control connection 565. For example, a REWIND command could be sent from the client 560 to the server 564 as an HTTP packet 570 of the form: "POST/HTTP/1.0<Entity-Body containing rewind command in any suitable media protocol>". The server can answer to this request with an HTTP response 572 of the form: "HTTP/1.0 200ok<Entity-Body containing rewind response in any suitable media protocol>".

The HTTP protocol can be also used to send media data across firewalls. The client can send a GET request to the video server, and the video server can then send the video data as the Entity-Body of the HTTP response to this GET request.

Some firewalls may be restrictive with respect to HTTP data and may permit HTTP packets to traverse only on a certain port, e.g., port 80 and/or port 8080. Fig. 5C illustrates one such situation. In this case, the control and data communications for the various data stream, e.g., audio, video, and/or annotation associated with different rendering sessions (and different clients) may be multiplexed using conventional multiplexer code and/or circuit 506 at client application 300 prior to being sent via port 502 (which may represent, for example, HTTP port 80 or HTTP port 8080). The inventive combined use of the HTTP protocol and of the multiplexer for transmitting media control and data is referred to as the HTTP multiplex protocol, and can be used to send this data across firewalls that only allow HTTP traffic on specific ports, e.g., port 80 or 8080.

At server 402, representing, for example, server 104 of Fig. 1, conventional demultiplexer code and/or circuit 508 may be employed to decode the received data packets to identify which stream the control request is associated with. Likewise, data sent from server 402 to client application 300 may be multiplexed in advance at server 402 using for example conventional multiplexer code and/or circuit 510. The multiplexed data is then sent via port 502. At client application 300, the multiplexed data may be decoded via conventional demultiplexer code and/or circuit 512 to identify which stream the received data packets is associated with (audio, video, or annotation).

Multiplexing and demultiplexing at the client and/or server may be facilitated for example by the use of the Request-URL part of the Request-Line of HTTP requests. As mentioned above, the structure of HTTP requests is described in RFC 1945. The Request-URL may, for example, identify the stream associated with the data and/or control request being transmitted. In one embodiment, the additional information in the Request-URL in the HTTP header may be as small as one or a few bits added to the HTTP request sent from client application 300 to server 402.

To further facilitate discussion of the inventive HTTP multiplexing technique, reference may now be made to Fig. 5D. In Fig. 5D, the plug-in application 660 within client plug-in application 660 may attempt to receive media data (e.g., video, audio, annotation, or the like) by first sending a control request 670 to server 664 via control connection 665. The control request is an HTTP request, which arrives at the HTTP default port 654 on server 664. As mentioned earlier, the default HTTP port may be either port 80 or port 8080 in one embodiment.

In one example, the control request 670 from client plug-in 660 takes the form of a command to "POST/12469 HTTP/1.0<Entity-Body>" which indicates to the server (through the designation 12469 as the Request-URL) that this is a control connection. The Entity-Body contains, as described above, binary data that informs the video server that the client plug-in 660 wants to display a certain video or audio clip. Software codes within server 664 may be employed to assign a unique ID to this particular request from this particular client.

For discussion sake, assume that server 664 associates unique ID 35,122 with a video data connection between itself and client plug-in application 660, and unique ID 29,999 with an audio data connection between itself and client plug-in application. The unique ID is then communicated as message 672 from server 664 to client plug-in application 660, again through the aforementioned HTTP default port using data connection 667. The Entity-Body of message 672 contains, among other things and as depicted in detail 673, the audio and/or video session ID. Note that the unique ID is unique to each data connection (e.g., each of the audio, video, and annotation connections) of each client plug-in application (since there may be multiple client plug-in applications attempting to communicate through the same port).

Once the connection is established, the same unique ID number is employed by the client to issue HTTP control requests to server 664. By way of example, client plug-in application 660 may issue a command "GET/35,122 HTTP/1.0" or "POST/35,122 HTTP/1.0<Entity-Body containing binary data with the REWIND media command>" to request a video file or to rewind on the video file. Although the rewind command is used in Figs. 5A-5D to facilitate ease of discussion, other media commands, e.g., fast forward, pause, real-time play, live-play, or the like, may of course be sent in the Entity-Body. Note that the unique ID is employed in place of or in addition to the Request-URL to qualify the Request-URL.

Once the command is received by server 664, the unique ID number (e.g., 35,122) may be employed by the server to demultiplex the command to associate the command with a particular client and data file. This unique ID number can also attach to the HTTP header of HTTP responses set from server 664 to client plug-in application 660, through the same HTTP default port 654 on server 664, to permit client plug-in application 660 to ascertain whether an HTTP data packet is associated with a given data stream.

Advantageously, the invention permits media control commands and media data to be communicated between the client computer and the server computer via the default HTTP port, e.g., port 80 or 8080 in one embodiment. even if HTTP packets are otherwise blocked by a firewall disposed between the client computer and the server computer. The association of each control connection and data connection to each client with a unique ID advantageously permits multiple control an data connections (from one or more clients) to be established through the same default HTTP port on the server, advantageously bypassing the firewall. Since both the server and the client have the demultiplexer code and/or circuit that resolve a particular unique ID into a particular data stream, multiplexed data communication is advantageously facilitated thereby.

In some networks, it may not be possible to traverse the firewall due to stringent firewall policies. As mentioned earlier, it may be possible in these situations to allow the client application to communicate with a server using a proxy. Fig. 6 illustrates this situation wherein client application 300 employs proxy 602 to communicate with server 402. The use of proxy 602 may be necessary since client application 300 may employ a protocol which is strictly prohibited by firewall 604. The identity of proxy 602 may be found in browser program 306, e.g., Netscape as it employs the proxy to download its web pages, or may be configured by the user himself. Typical protocols that may employ a proxy for communication, e.g., proxy 602, includes HTTP and UDP.

In accordance with one embodiment of the present invention, the multiple protocols that may be employed for communication between a server computer and a client computer are tried in parallel during autodetect. In other words, the connections depicted in Figs. 3, 4, 5C and 6 may be attempted simultaneously and in parallel over different control connections by the client computer. Via these control connections, the server is requested to respond with various protocols.

If the predefined "best" protocol (predetermined in accordance with some predefined protocol priority) is received by the client application from the server, autodetect may, in one embodiment, end immediately and the "best" protocol is selected for immediate communication. In one real-time data rendering application, UDP is considered the "best" protocol, and the receipt of UDP data by the client may trigger the termination of the autodetect.

If the "best" protocol has not been received after a predefined time period, the most advantageous protocol (in terms of for example data transfer rate and/or transmission control) is selected among the set of protocols received by the client. The selected protocol may then be employed for communication between the client and the server.

Fig. 7 depicts, in accordance with one embodiment of the present invention, a simplified flowchart illustrating the steps of the inventive autodetect technique. In Fig. 7, the client application starts (in step 702) by looking up the HTTP proxy, if there is any, from the browser. As stated earlier, the client computer may have received a web page from the browser, which implies that the HTTP protocol may have been employed by the browser program for communication. If an HTTP proxy is required, the name and location of the HTTP proxy is likely known to the browser, and this knowledge may be subsequently employed by the client to at least enable communication with the server using the HTTP proxy protocol, i.e., if a more advantageous protocol cannot be ascertained after autodetect.

In step 704, the client begins the autodetect sequence by starting in parallel the control thread 794, along with five protocol threads 790, 792, 796, 798, and 788. As the term is used herein, parallel refers to both the situation wherein the multiple protocol threads are sent parallely starting at substantially the same time (having substantially similar starting time), and the situation wherein the multiple protocol threads simultaneously execute (executing at the same time), irrespective when each protocol thread is initiated. In the latter case, the multiple threads may have, for example, staggered start time and the initiation of one thread may not depend on the termination of another thread.

Control thread 794 represents the thread for selecting the most advantageous protocol for communication. The other protocol threads 790, 792, 796, 798, and 788 represent threads for initiating in parallel communication using the various protocols, e.g., UDP, TCP, HTTP proxy, HTTP through port 80 (HTTP 80), and HTTP through port 8080 (HTTP 8080). Although only five protocol threads are shown, any number of protocol threads may be initiated by the client, using any conventional and/or suitable protocols. The steps associated with each of threads 794, 790, 792, 796, 798, and 788 are discussed herein in connection with Figs. 8A-8E and 9.

In Fig. 8A, the UDP protocol thread is executed. The client inquires in step 716 whether a UDP proxy is required. If the UDP proxy is required, the user may obtain the name of the UDP proxy from, for example, the system administrator in order to use the UDP proxy to facilitate communication to the proxy (in step 718). If no UDP proxy is required, the client may directly connect to the server (in step 720). Thereafter, the client may begin sending a data request (i.e., a control request) to the server in step 722 using the UDP protocol (either through the proxy if a proxy is involved or directly to the server if no proxy is required).

In Fig. 8B, the TCP protocol thread is executed. If TCP protocol is employed, the client typically directly connects to the server (in step 726). Thereafter, the client may begin sending a data request (i.e., a control request) to the server using the TCP protocol (step 724).

In Fig. 8C, the HTTP protocol thread is executed. The client inquires in step 716 whether an HTTP proxy is required. If the HTTP proxy is required, the user may obtain the name of the HTTP proxy from, for example, the browser since, as discussed earlier, the data pertaining to the proxy may be kept by the browser. Alternatively, the user may obtain data pertaining to the HTTP proxy from the system administrator in order to use the HTTP proxy to facilitate communication to the server (in step 732).

If no HTTP proxy is required, the client may directly connect to the server (in step 730). Thereafter, the client may begin sending a data request (i.e., a control request) to the server in step 734 using the HTTP protocol (either through the proxy if a proxy is involved or directly to the server if no proxy is required).

In Fig. 8D, the HTTP 80 protocol thread is executed. If HTTP 80 protocol is employed, HTTP data may be exchanged but only through port 80, which may be for example the port on the client computer through which communication with the network is permitted. Through port 80, the client typically directly connects to the server (in step 736). Thereafter, the client may begin sending a data request (i.e., a control request) to the server (step 738) using the HTTP 80 protocol.

In Fig. 8E, the HTTP 8080 protocol thread is executed. If HTTP 8080 protocol is employed, HTTP data may be exchanged but only through port 8080, which may be the port on the client computer for communicating with the network. Through port 8080, the client typically directly connects to the server (in step 740). Thereafter, the client may begin sending a data request (i.e., a control request) to the server (step 742) using the HTTP 8080 protocol. The multiplexing and demultiplexing techniques that may be employed for communication through port 8080, as well as port 80 of Fig. 8D, have been discussed earlier and are not repeated here for brevity sake.

Fig. 9 illustrates, in accordance with one embodiment of the present invention, control thread 794 of Fig. 7. It should be emphasized that Fig. 7 is but one way of implementing the control thread; other techniques of implementing the control thread to facilitate autodetect should be apparent to those skilled in the art in view of this disclosure. In step 746, the thread determines whether the predefined timeout period has expired. The predefined timeout period may be any predefined duration (such as 7 seconds for example) from the time the data request is sent out to the server (e.g., step 722 of Fig. 8A). In one embodiment, each protocol thread has its own timeout period whose expiration occurs at the expiration of a predefined duration after the data request using that protocol has been sent out. When all the timeout periods associated with all the protocols have been accounted for, the timeout period for the autodetect technique is deemed expired.

If the timeout has occurred, the thread moves to step 754 wherein the most advantageous protocol among the set of protocols received back from the server is selected for communication. As mentioned, the selection of the most advantageous protocol may be performed in accordance with some predefined priority scheme, and data regarding the selected protocol may be saved for future communication sessions between this server and this client.

If no timeout has occurred, the thread proceeds to step 748 to wait for either data from the server or the expiration of the timeout period. If timeout occurs, the thread moves to step 754, which has been discussed earlier. If data is received from the server, the thread moves to step 750 to ascertain whether the protocol associated with the data received from the server is the predefined "best" protocol, e.g., in accordance with the predefined priority.

If the predefined "best" protocol (e.g., UDP in some real-time data rendering applications) is received, the thread preferably moves to step 754 to terminate the autodetect and to immediately begin using this protocol for data communication instead of waiting for the timeout expiration. Advantageously, the duration of the autodetect sequence may be substantially shorter than the predefined timeout period. In this manner, rapid autodetect of the most suitable protocol and rapid establishment of communication are advantageously facilitated.

If the predefined "best" protocol is not received in step 750, the thread proceeds to step 752 to add the received protocol to the received set. This received protocol set represents the set of protocols from which the "most advantageous" (relatively speaking) protocol is selected. The most advantageous protocol is ascertained relative to other protocols in the received protocol set irrespective whether it is the predefined "best" protocol in accordance with the predefined priority. As an example of a predefined protocol priority, UDP may be deemed to be best (i.e., the predefined best), followed by TCP, HTTP, then HTTP 80 and HTTP 8080 (the last two may be equal in priority). As mentioned earlier, the most advantageous protocol is selected from the received protocol set preferably upon the expiration of the predefined timeout period.

From step 752, the thread returns to step 746 to test whether the timeout period has expired. If not, the thread continues along the steps discussed earlier.

Note that since the invention attempts to establish communication between the client application and the server computer in parallel, the time lag between the time the autodetect mechanism begins to execute and the time when the most advantageous protocol is determined is minimal. If communication attempts have been tried in serial, for example, the user would suffer the delay associated with each protocol thread in series, thereby disadvantageously lengthening the time period between communication attempt and successful establishment of communication.

The saving in time is even more dramatic in the event the network is congested or damaged. In some networks, it may take anywhere from 30 to 90 seconds before the client application realizes that an attempt to connect to the server (e.g., step 720, 726, 730, 736, or 740) has failed. If each protocol is tried in series, as is done in one embodiment, the delay may, in some cases, reach minutes before the user realizes that the network is unusable and attempts should be made at a later time.

By attempting to establish communication via the multiple protocols in parallel, network-related delays are suffered in parallel. Accordingly, the user does not have to wait for multiple attempts and failures before being able to ascertain that the network is unusable and an attempt to establish communication should be made at a later time. In one embodiment, once the user realizes that all parallel attempts to connect with the network and/or the proxies have failed, there is no need to make the user wait until the expiration of the timeout periods of each thread. In accordance with this embodiment, the user is advised to try again as soon as it is realized that parallel attempts to connect with the server have all failed. In this manner, less of the user's time is needed to establish optimal communication with a network.

While this invention has been described in terms of several preferred embodiments, there are alterations, permutations, and equivalents which fall within the scope of this invention. For example, although the invention has been described with reference to sending out protocol threads in parallel, the automatic protocol detection technique also applies when the protocol threads are sent serially. In this case, while it may take longer to select the most advantageous protocol for selection, the automatic protocol detection technique accomplishes the task without requiring any sophisticated technical knowledge on the part of the user of the client computer. The duration of the autodetect technique, even when serial autodetect is employed, may be shortened by trying the protocols in order of their desirability and ignoring less desirable protocols once a more desirable protocol is obtained. It should also be noted that there are many alternative ways of implementing the methods and apparatuses of the present invention. It is therefore intended that the following appended claims be interpreted as including all such alterations, permutations, and equivalents as fall within the true spirit and scope of the present invention.

What is claimed is:

1. In a computer network, a method for automatically detecting a most advantageous protocol for communication by a client computer, said client computer being configured to be coupled to a server computer via a computer network, comprising:

sending, from said client computer to said server computer, a plurality of data requests, each of said data requests employing a different protocol and a different connection, said data requests being configured to solicit, responsive to said data requests, a set of responses from said server computer, each of said responses employing a protocol associated with a respective one of said data requests;

receiving at said client computer at least a subset of said responses; and

monitoring said subset of said responses as each response is received from said server computer to select said most advantageous protocol from protocols associated with said subset of said responses, wherein said most advantageous protocol is determined based on a predefined protocol priority.

2. The method of claim 1 wherein said sending is performed in a substantially parallel manner.

3. The method of claim 2 wherein said plurality of data requests are sent substantially at the same time.

4. The method of claim 2 wherein said plurality of data requests execute concurrently.

5. The method of claim 2 wherein said most advantageous protocol represents a predefined best protocol, said client computer, upon receiving a response employing said predefined best protocol, immediately designates said predefined best protocol said most advantageous protocol.

6. The method of claim 3 wherein said control thread selects said most advantageous protocol upon an expiration of a timeout period if said predefined best protocol is not received at said client computer upon said expiration, said timeout period being measured from a transmitting time of said one of said data requests, said subset of responses representing responses received from said server computer during said timeout period.

7. The method of claim 4 wherein said client computer represents a computer executing an application for rendering real-time data as said real-time data is received from said server computer.

8. The method of claim 7 wherein said computer network represents the Internet and said predefined best protocol represents UDP.

9. The method of claim 8 wherein said real-time data represents one of a video data stream, an audio data stream, and an annotation data stream.

10. In a computer network, a method for automatically detecting a most advantageous protocol for communication by a client computer, said client computer being configured to be coupled to a server computer via a computer network, comprising:

sending, from said client computer to said server computer, a plurality of data requests, each of said data requests employing a different protocol and a different connection;

receiving at least a subset of said data requests at said server computer;

sending a set of responses from said server computer to said client computer, said set of responses being responsive to said subset of said data requests, each of said responses employing a protocol associated with a respective one of said subset of said data requests;

receiving at said client computer at least a subset of said responses; and selecting, for said communication between said client computer and said server computer, said most advantageous protocol from protocols associated with said subset of said responses, wherein said most advantageous protocol is determined based on a predefined protocol priority.

11. The method of claim 10 wherein said sending is performed in a substantially parallel manner.

12. The method of claim 10 wherein said most advantageous protocol represents a predefined best protocol, said selecting comprises:

monitoring, employing said client computer, said subset of said responses as each one of said subset of said responses is received at said client computer from said server computer for a response employing said predefined best protocol; and

designating said predefined best protocol said most advantageous protocol, thereby immediately permitting said client computer to employ said predefined best protocol for communication with said server computer without further receiving additional responses from said server computer.

13. The method of claim 10 wherein said selecting comprises:

selecting at an expiration of a timeout period said most advantageous protocol from said protocols associated with said subset of responses, said subset of responses representing responses received from said server computer during said timeout period.

14. The method of claim 13 wherein said timeout period represents a predefined time period associated with one of said data requests measured form a transmitting time of said one of said data requests.

15. The method of claim 14 wherein said client computer represents a computer executing an application for rendering real-time data as said real-time data is received from said server computer.

16. The method of claim 15 wherein said computer network represents the Internet and said predefined best protocol represents UDP.

17. The method of claim 15 wherein said real-time data represents one of a video data stream, an audio data stream, and an annotation data stream.

18. The method of claim 17 wherein said data requests employ at least one of a UDP, TCP, HTTP proxy, HTTP 80, and HTTP 8080 protocols.

19. The method of claim 18 wherein said HTTP 80 protocol represents a protocol for permitting multiple HTTP data streams to be transmitted in a multiplexed manner through port 80, said multiple HTTP data streams representing said video data stream and said audio data stream.

20. The method of claim 19 wherein said HTTP 8080 protocol represents a protocol for permitting multiple HTTP data streams to be transmitted in a multiplexed manner through port 8080, said multiple HTTP data streams representing said video data stream and said audio data stream.

21. A computer-readable medium containing computer-readable instructions for automatically detecting a most advantageous protocol for communication by a client computer, said client computer being configured for coupling to a server computer via a computer network, said computer-readable instructions comprise:

computer-readable instructions for sending in a substantially parallel manner, from said client computer to said server computer, a plurality of data requests, each of said data requests employing a different protocol and a different connection, said data requests being configured to solicit, responsive to said data requests, a set of responses from said server computer, each of said responses employing a protocol associated with a respective one of said data requests;

computer-readable instructions for receiving at said client computer at least a subset of said responses; and

computer-readable instructions for monitoring said subset of said responses as each response is received from said server computer to select said most advantageous protocol from protocols associated with said subset of said responses, wherein said most advantageous protocol is determined based on a predefined protocol priority.

22. The computer-readable medium of claim 21 wherein said most advantageous protocol represents a predefined best protocol, said client computer, upon receiving a response employing said predefined best protocol, immediately designates said predefined best protocol said most advantageous protocol.

23. The computer-readable medium of claim 22 wherein said control thread selects said most advantageous protocol upon an expiration of a timeout period if said predefined best protocol is not received at said client computer upon said expiration, said timeout period being measured from a transmitting time of said one of said data requests, said subset of responses representing responses received from said server computer during said timeout period.

24. The computer-readable medium of claim 23 wherein said client computer represents a computer executing an application for rendering real-time data as said real-time data is received from said server computer.

25. The computer-readable medium of claim 24 wherein said computer network represents the Internet and said predefined best protocol represents UDP.

26. The computer-readable medium of claim 25 wherein said real-time data represents one of a video data stream, an audio data stream, and an annotation data stream.

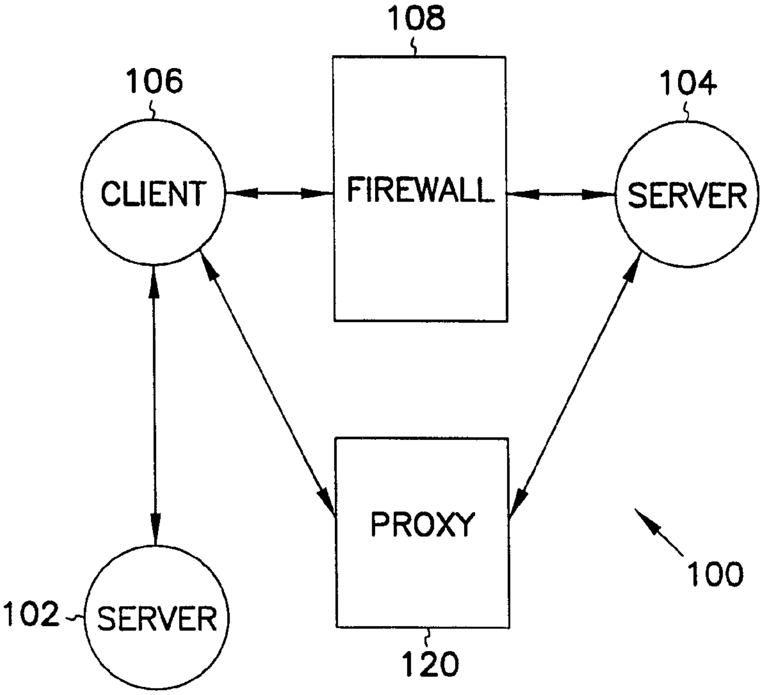


FIG. 1

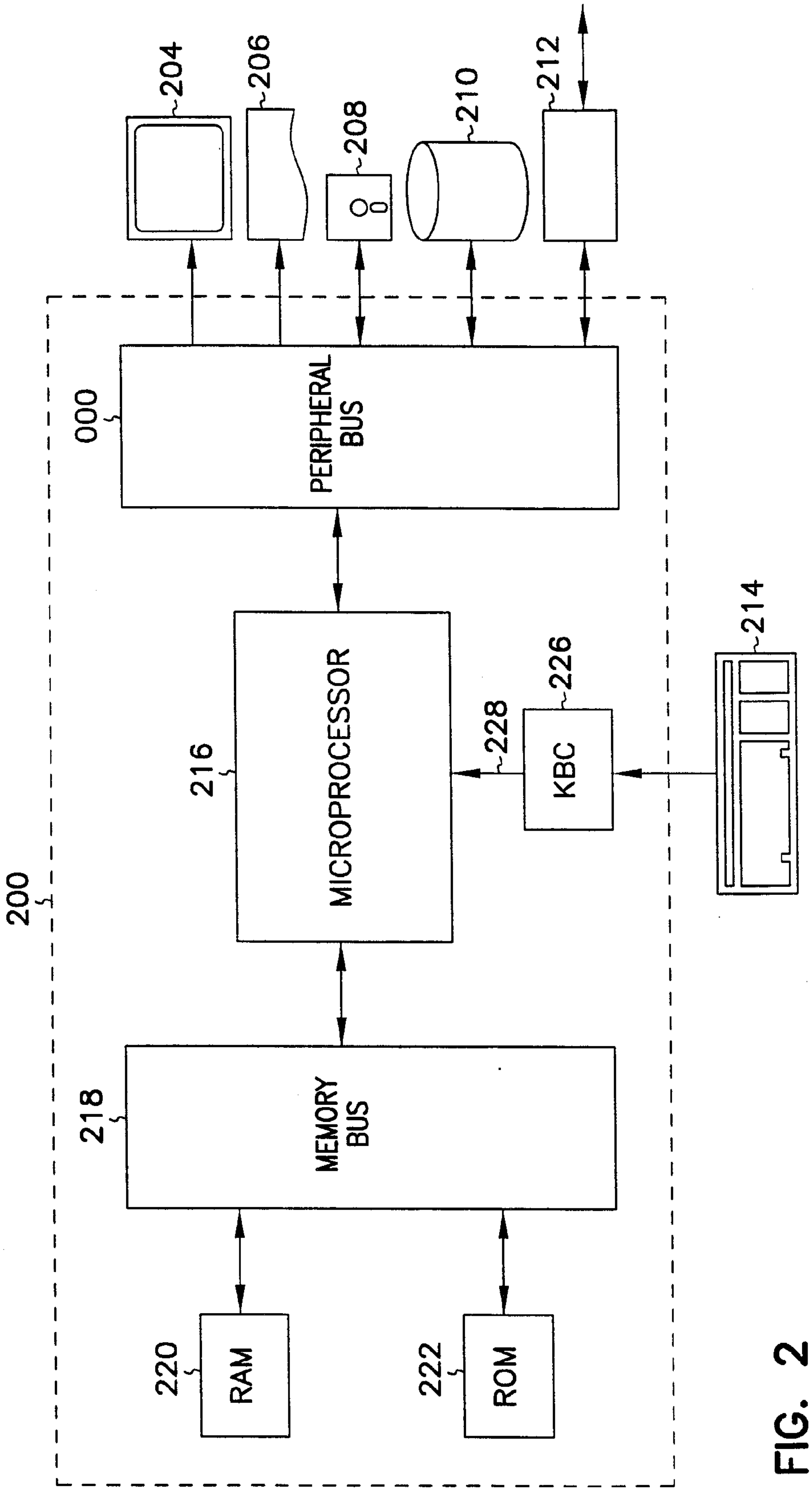

204
206
208
210
212
000
PERIPHERAL BUS
216
MICROPROCESSOR
200
228
226
KBC
214
218
MEMORY BUS
220
RAM
222
ROM

FIG. 2

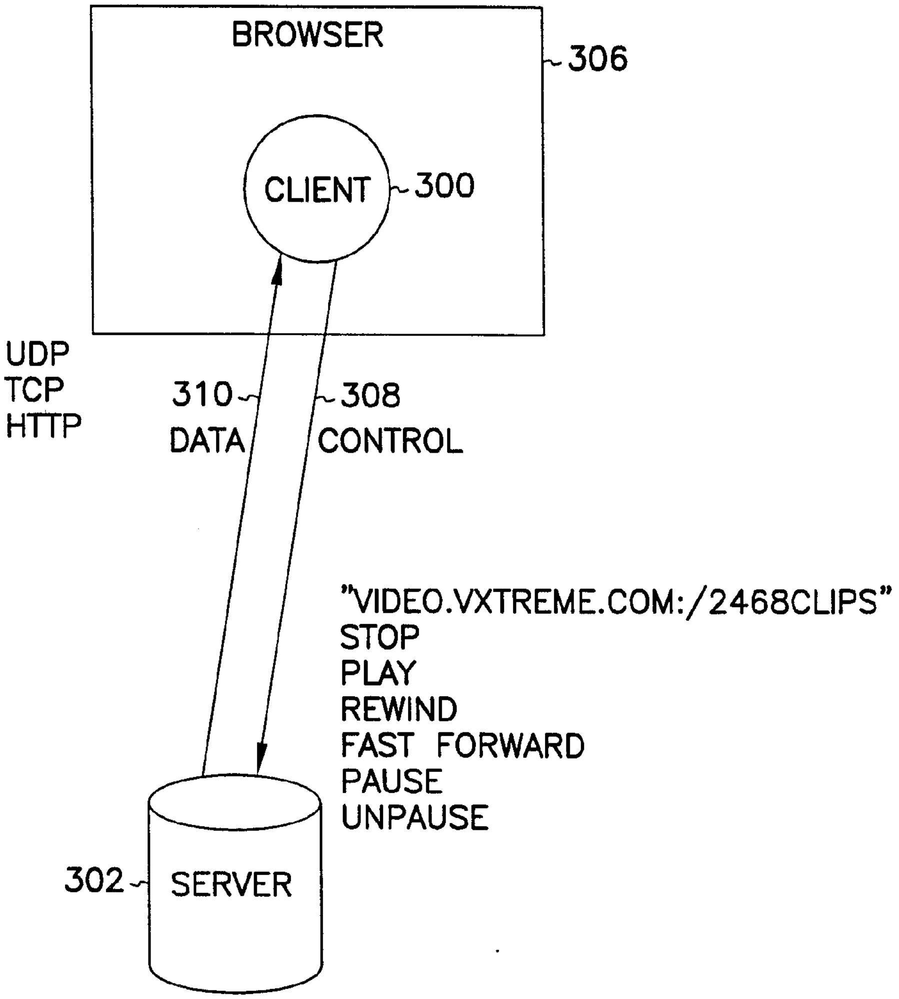

BROWSER
306
CLIENT
300
UDP
TCP
HTTP
310
DATA
308
CONTROL
"VIDEO.VXTREME.COM:/2468CLIPS"
STOP
PLAY
REWIND
FAST FORWARD
PAUSE
UNPAUSE
302
SERVER

FIG. 3

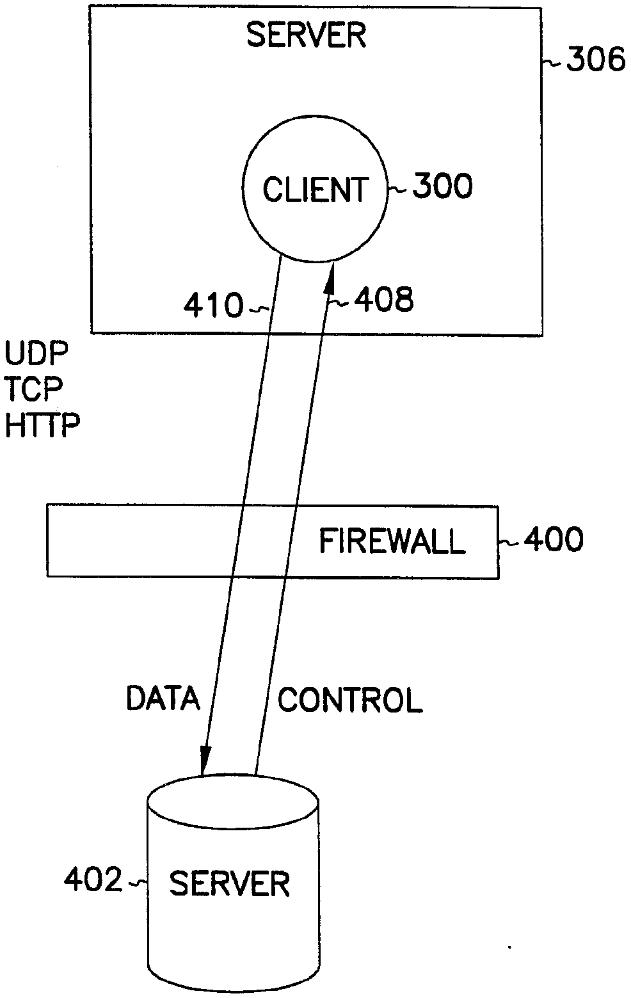

SERVER
306
CLIENT
300
410
408
UDP
TCP
HTTP
FIREWALL
400
DATA
CONTROL
402
SERVER

FIG. 4

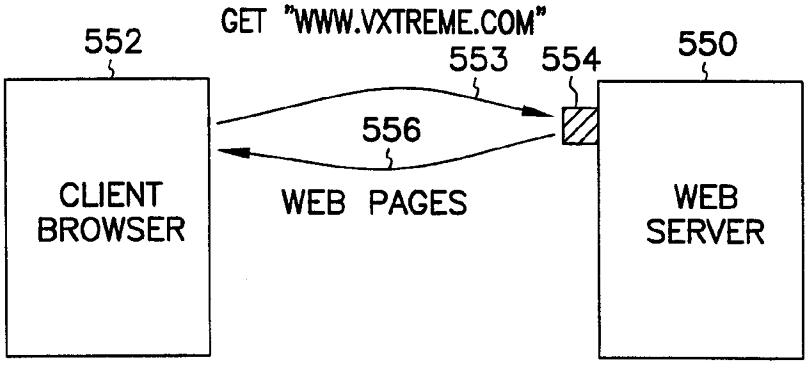


FIG. 5A (PRIOR ART)

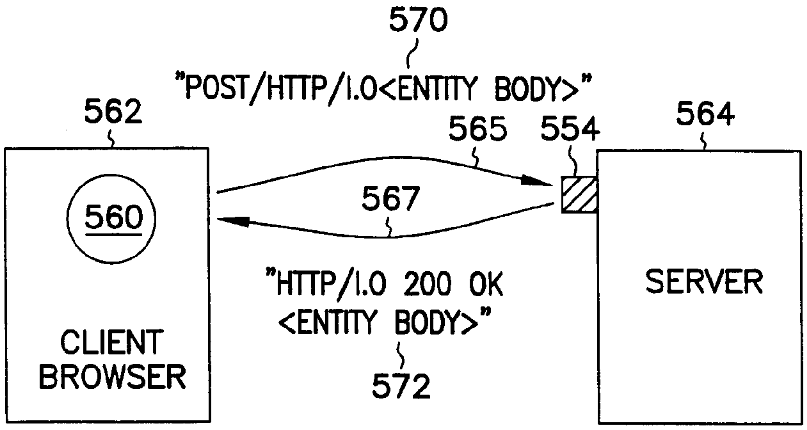


FIG. 5B

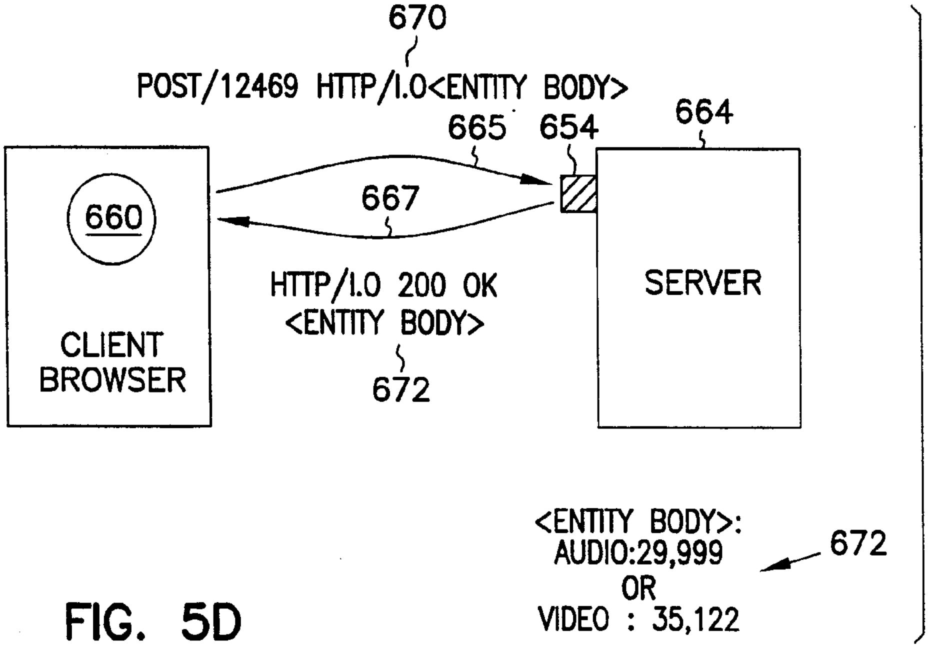


FIG. 5D

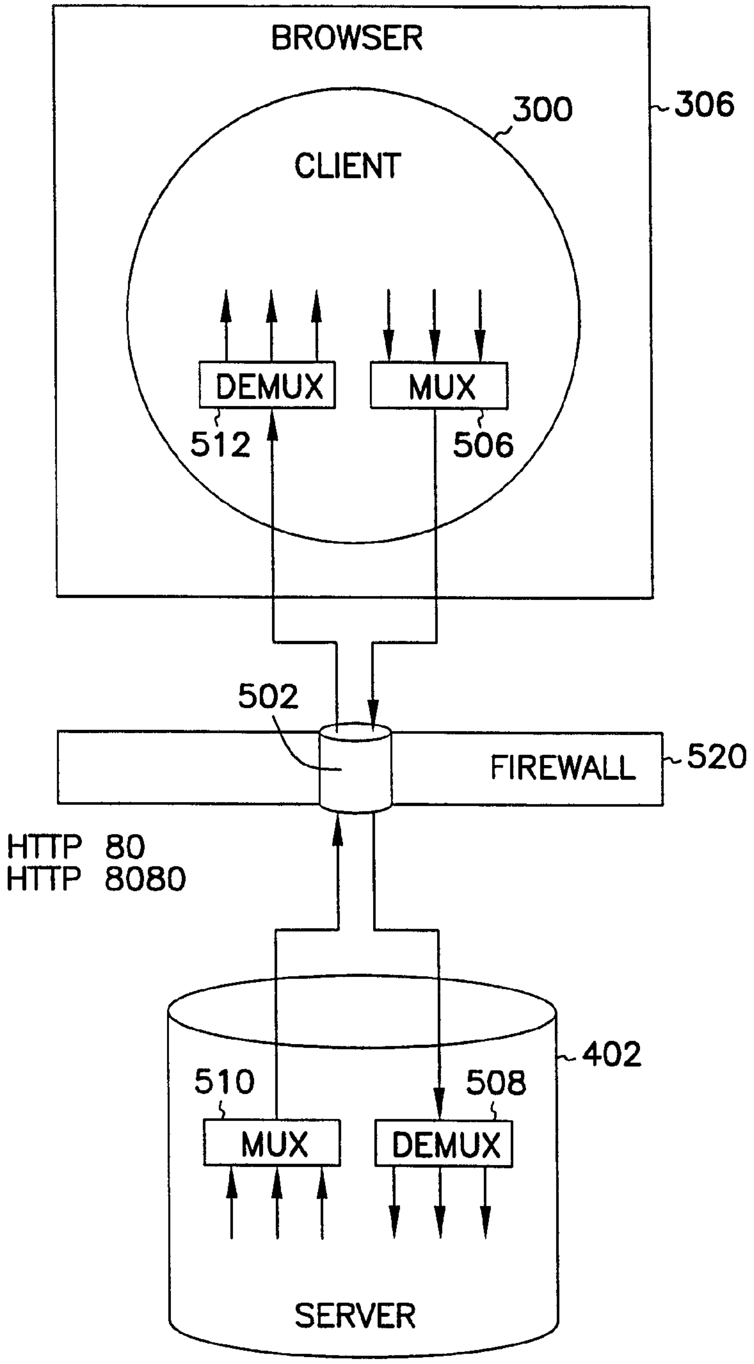

BROWSER
CLIENT
300
306
DEMUX
MUX
512
506
502
FIREWALL
520
HTTP 80
HTTP 8080
402
510
508
MUX
DEMUX
SERVER

FIG. 5C

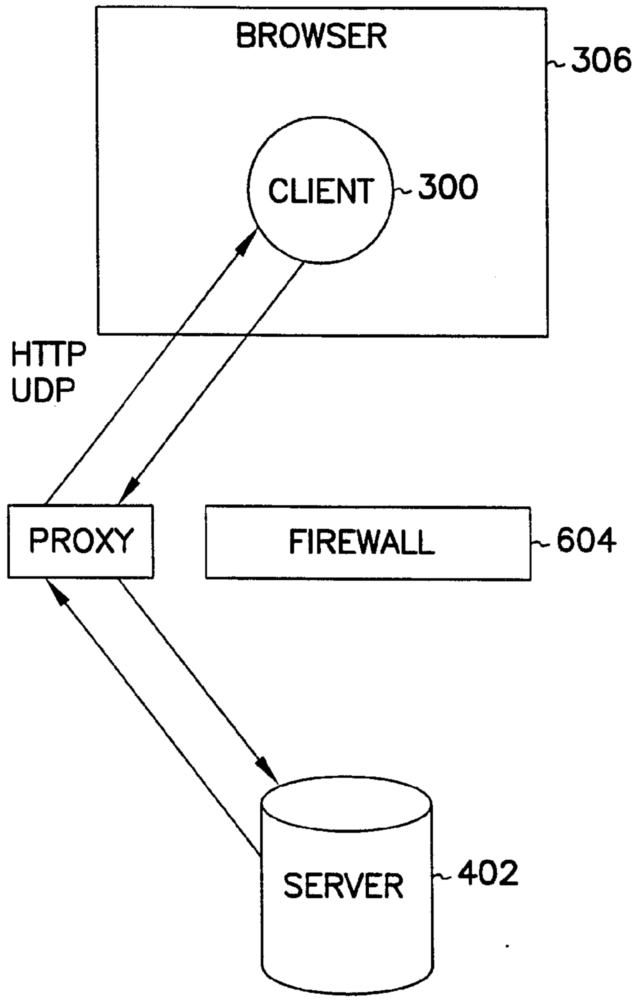

BROWSER
306
CLIENT
300
HTTP
UDP
PROXY
FIREWALL
604
SERVER
402

FIG. 6

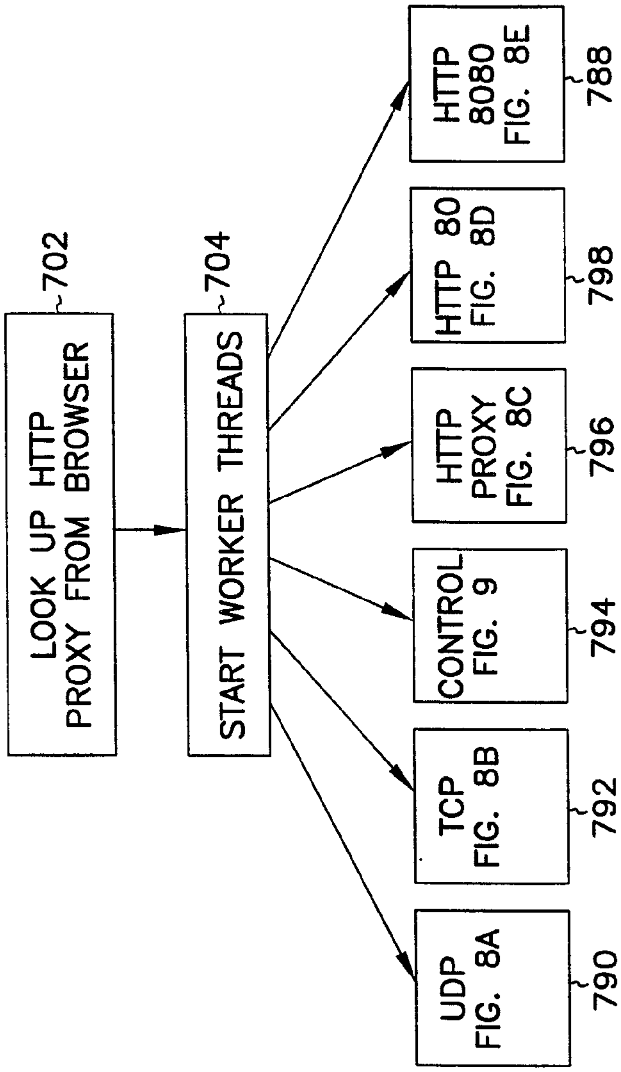

LOOK UP HTTP PROXY FROM BROWSER
702
START WORKER THREADS
704
UDP FIG. 8A
790
TCP FIG. 8B
792
CONTROL FIG. 9
794
HTTP PROXY FIG. 8C
796
HTTP 80 FIG. 8D
798
HTTP 8080 FIG. 8E
788

FIG. 7

FIG. 8A

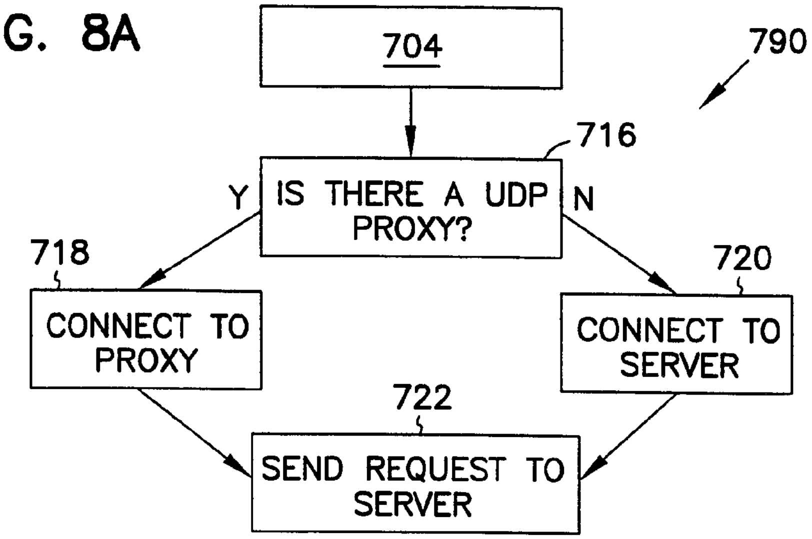


FIG. 8B

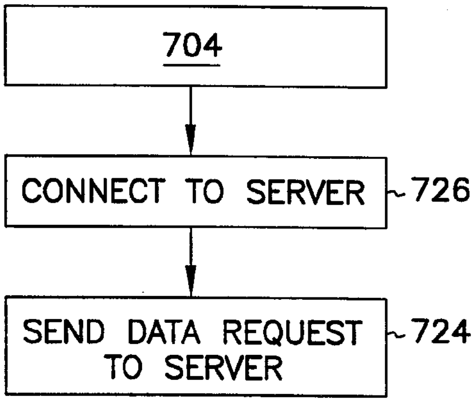


FIG. 8C

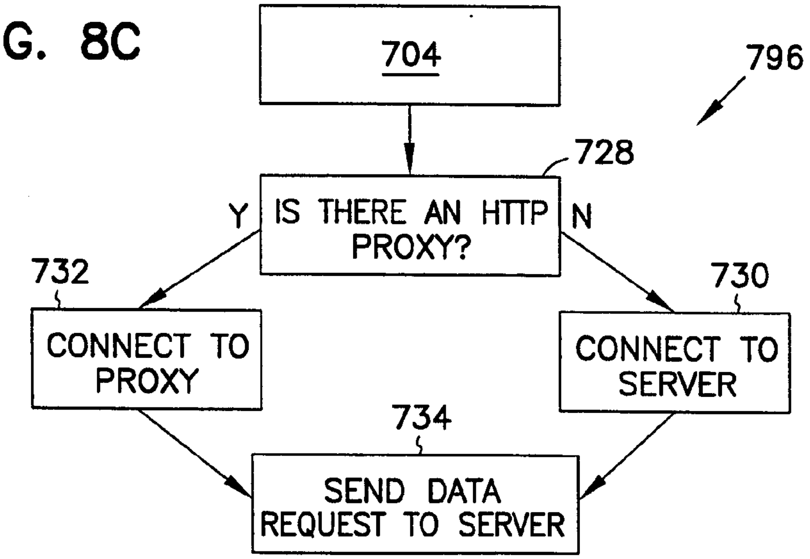


FIG. 8D

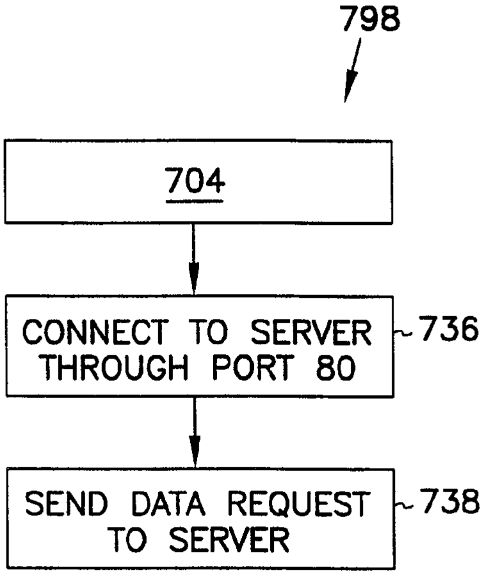
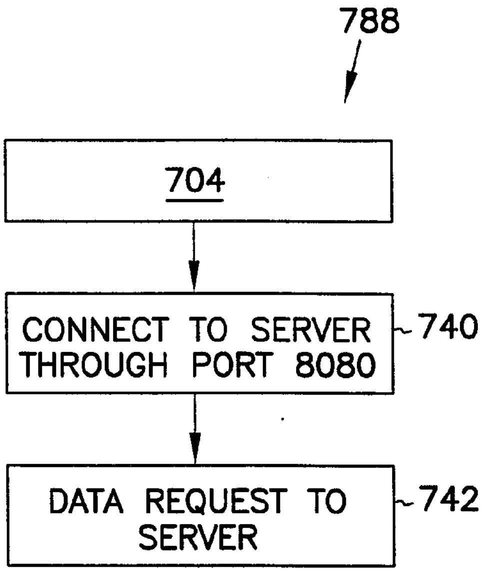

798
704
CONNECT TO SERVER THROUGH PORT 80
736
SEND DATA REQUEST TO SERVER
738

FIG. 8E


788
704
CONNECT TO SERVER THROUGH PORT 8080
740
DATA REQUEST TO SERVER
742

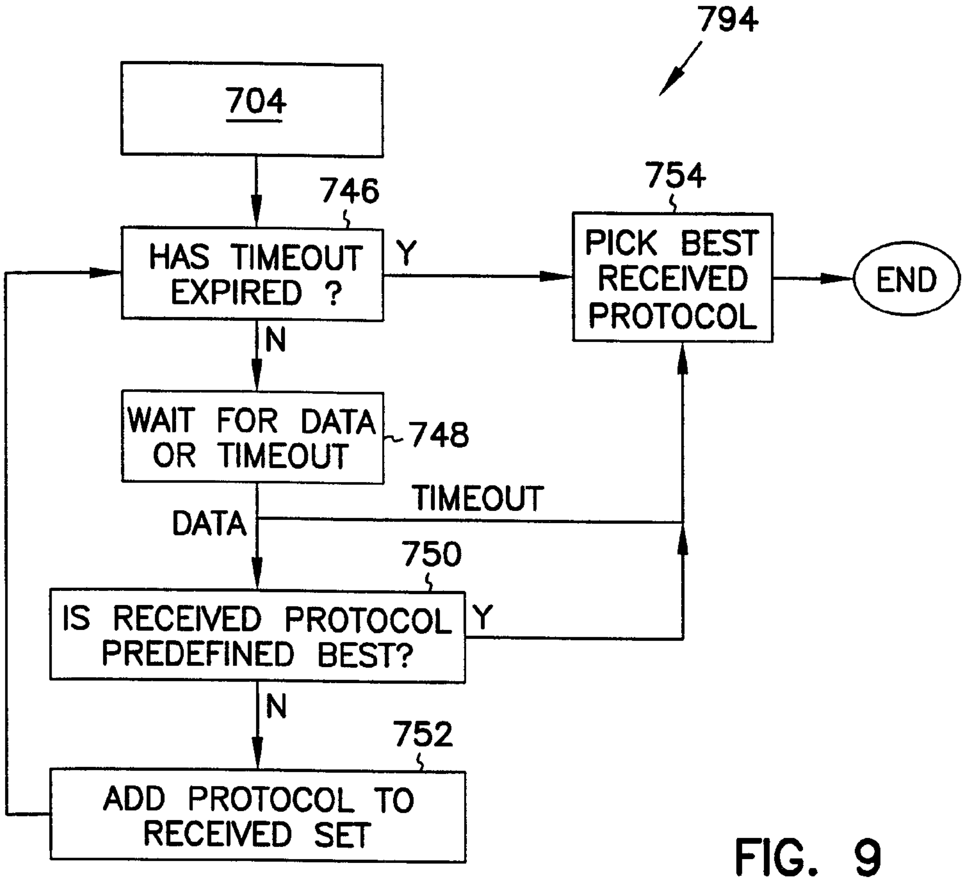

794
704
746
HAS TIMEOUT EXPIRED ?
Y
N
754
PICK BEST RECEIVED PROTOCOL
END
WAIT FOR DATA OR TIMEOUT
748
TIMEOUT
DATA
750
IS RECEIVED PROTOCOL PREDEFINED BEST?
Y
N
752
ADD PROTOCOL TO RECEIVED SET

FIG. 9

INTERNATIONAL SEARCH REPORT

Inte onal Application No
PCT/US 98/01797

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04L29/06 H04L29/08

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 613 274 A (IBM) 31 August 1994 see abstract see page 2, line 54 - line 55 see page 3, line 7 - line 11 ---	1,10,12
A	EP 0 751 656 A (CANON KK) 2 January 1997 see abstract see page 2, column 2, line 27 - line 40 see page 4, column 6, line 52 - line 57 see page 5, column 7, line 22 - column 8, line 13 see figure 2 see figure 3 see page 2, column 1, line 24 - line 27 see page 2, column 1, line 52 - column 2, line 2 -----	1-26

□ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search 8 June 1998	Date of mailing of the international search report 17/06/1998
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Adkhis, F

INTERNATIONAL SEARCH REPORT

ınformation on patent family members

Inte onal Application No

PCT/US 98/01797

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0613274 A	31-08-1994	US 5537417 A	16-07-1996
		JP 7049823 A	21-02-1995
		JP 8001622 B	10-01-1996
EP 0751656 A	02-01-1997	US 5710908 A	20-01-1998
		JP 9062593 A	07-03-1997

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : 0 613 274 A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number : 94300677.5

(22) Date of filing : 28.01.94

(51) Int. Cl.^{5} : H04L 29/06

(30) Priority : 29.01.93 US 15366

(43) Date of publication of application :
31.08.94 Bulletin 94/35

(84) Designated Contracting States :
DE FR GB

(71) Applicant : International Business Machines Corporation
Old Orchard Road
Armonk, N.Y. 10504 (US)

(72) Inventor : Sharma, Mohan Byrappa
12148 Jollyvill Road No.316
Austin, Texas 78759 (US)
Inventor : Yeung, Yue
11714 D-K Ranch Road
Austin, Texas 78759 (US)
Inventor : Cheng, Chungsiang
10629 Floral Drive
Austin, Texas 78759 (US)

(74) Representative : Moss, Robert Douglas
IBM United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester Hampshire SO21 2JN (GB)

(54) Socket structure for concurrent multiple protocol access.

(57) In a multiprotocol environment, a new socket structure is provided which moves the decision on which protocol to use until the time that the connection is actually made between nodes in the network. The new socket structure is created for every endpoint. All the protocols which could potentially be used to send or receive data is sent a request to create a protocol control block at the time the new socket is created. A selection process determines which of the protocols could be used by the endpoint. The new socket then contains information on each selected protocol. At the time a connection is established, any of the selected protocols could be used. The choice of which protocol to use can be based on user preferences, which protocols are available, the name of the service unit.

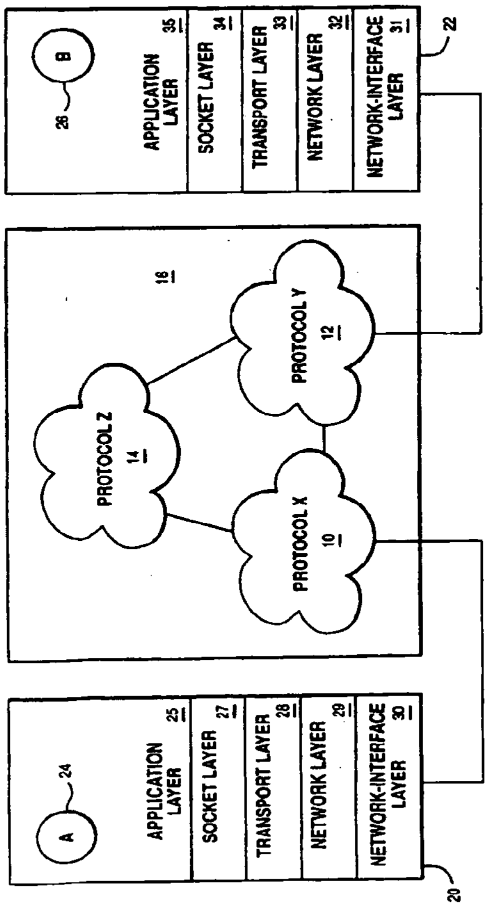


FIG. 1

EP 0 613 274 A2

Jouve, 18, rue Saint-Denis, 75001 PARIS

BACKGROUND OF THE INVENTION

This invention relates generally to data communication in a network of computer systems. More particularly, it relates to a communication endpoint structure which enables application programs resident on systems to communicate through such a network irrespective of the protocol for which the application was written and the protocols available on the network.

Once upon a time, almost all computer systems were standalone processors to which private peripheral devices such as displays, printers and disk drives were connected, acting independently from any other computer system. Yet, it became apparent that there were substantial gains to be made if several computer systems could be cooperatively coupled together. Today, it has become commonplace to couple a multitude of computer systems into a distributed environment such as a local area or wide area network.

However, there are many vendors who have developed their own hardware and software solutions for integrating multiple computer systems. In particular, they have developed different ideas of the format and protocols that should be followed in transporting data through the networks. Some protocols support expedited data transmission by bypassing the standard data flow controls; others require all data to go through the controls. Specialized protocols are used for transport tasks such as establishing and terminating connections between computer systems. Examples of well known communication protocols include System Network Architecture (SNA), Digital Network Architecture (DECnet), Transmission Control Protocol/Internet: Protocol (TCP/IP), Network Basic Input/Output System (NetBIOS), Open Systems Interconnection (OSI) and AppleTalk. Other protocols are known and widely used.

Most distributed application programs are written to an application programming interface (API) which assumes a particular communications protocol. However, the distributed environments which most organizations have assembled are quite complex, comprised of confederations of individual networks running different communication protocols. If the transport protocols assumed by the distributed application's API and the transport protocols actually implemented in one or more of the networks on which the organization would like to send its data are not the same, the use of the application is limited. The problems of such heterogeneity in communications protocols is expected to get worse as more organizations begin to communicate with each other through their networks to perform order processing, direct billing or other cross organization activities.

While the distributed applications could be rewritten so that they can run over each transport protocol or application gateways can be written for each distributed set of distributed applications, the cost of having programmers write all the necessary code makes these approaches economically unattractive. A preferred solution is presented in copending application, Serial No. 731,564, entitled "Compensation for Mismatched Transport Protocols in a Data Communications Network", by R.F. Bird et al, filed July 17, 1991 and hereby incorporated by reference. The incorporated application teaches a transport connection layer between a first transport user at one node in the network and a second transport user at a different node in the network. When the transport protocol assumed by the application at the first node is not available in the network, the data being transferred between the two nodes is automatically altered to be compatible with the available transport protocols. Thus, an organization is able to choose application programs solely on the basis of the functions they provide, rather than the protocols which they require.

The above referenced application teaches a generalized transport layer. One of the transport structures which is presently used in the Berkeley version of the UNIX(TM)environment is called a socket. A socket is an object that identifies a communication endpoint in a network. A socket hides the protocol of the network architecture beneath from the application. A socket allows the association of an endpoint, such as an application program, with one of the selected protocols. This association occurs when the endpoint is created. An endpoint creation implies a static association of the socket with the protocol, which will remain invariant. However, in a multiprotocol environment, as described in the above referenced application, which facilitates heterogeneous connectivity, a transport endpoint may need to be bound to any of several connectivity, a transport endpoint may need to be bound to any of several available protocols after the creation of the endpoint. Therefore, if sockets are to be used in such an environment, a new socket structure which allows dynamic association of the socket with the protocol must be devised. The present invention teaches such a socket structure.

Summary of the Invention

The present invention provides a system and a method for communicating between nodes in a computer network in which a plurality of protocols are useable by network nodes, the method comprising the steps of creating communication endpoint objects defining communication parameters for respective network nodes, the objects having information about the plurality of protocols which are useable by said node for inter-node communication; and at the time communication is requested between said nodes, establishing a connection between said nodes using a selected one of the plurality of protocols.

The present invention thereby facilitates late binding of an endpoint to a transport protocol in a distributed environment, enabling a more dynamic association between protocol and endpoint.

Preferably, the invention allows native access to a protocol by an application. It is also preferred that the invention provides the necessary information for non-native connections to a protocol.

The invention provides, in a preferred embodiment, a new socket structure which moves the decision on which protocol to use to the time that the connection is actually made between nodes in the network. In a multiprotocol network, the new socket structure can be created for every endpoint. For all of the protocols which could potentially be used to send or receive data a request is made to create a protocol control block at the time the new socket is created. A selection process determines which of the protocols could potentially be used by a given endpoint. The new socket for the endpoint then contains information on each of the selected protocols. At the time a connection is established, any of the selected protocols could be used. Native or non-native connections can be made. The choice of which protocol to use or the order of protocol preference can be based on user preferences through configuration, the local cache, or information from the named service unit.

Brief Description of the Drawings

The invention will now be described in more detail, by way of example, with reference to the accompanying drawings in which:

FIG. 1 is a diagram of two single protocol networks coupled together through a multiprotocol transport network.

FIG. 2 is a block diagram of the transport interface used according to an embodiment of the present invention.

FIG. 3 is a diagram of a conventional socket control block.

FIG. 4 is a diagram of the socket control block according to an embodiment of the present invention.

FIG. 5 is a flow diagram of the creation of a socket according to an embodiment of the present invention.

FIG. 6 is a flow diagram of establishing a connection in a multiprotocol transport network environment, using a multiprotocol socket according to the invention.

FIG. 7 is a representation of a computer system including system unit, keyboard, mouse and display.

FIG. 8 is a block diagram of the computer system components depicted in FIG. 7.

Detailed Description of the Drawings

The following description describes a socket based architecture. However, the invention is not limited to sockets and could be applied to similar communication endpoint objects in other operating systems.

Although the following description contains an adequate disclosure of conventional socket and network architecture to allow one skilled in the art to understand the present invention, the reader is referred to The Design and Implementation of the 4.3 BSD UNIX Operating System by S. J. Leffer et al., 1989 which is hereby incorporated by reference, for a more complete understanding of operating system based on the Berkley version of UNIXTM. Such operating systems are well known.

FIG. 1 shows three single protocol networks 10, 12, 14 which are interconnected through a gateway 59 built according to the principles of the present invention. With the growth of network distributed environments, it is not uncommon to see networks using four or five different protocols, such as TCP/IP, NetBIOS, SNA or AppleTALK. Since applications which run on one network will not often run with applications on the other, transport of data throughout the network is hindered. As discussed above, the MultiProtocol Transport Network (MPTN) 16 as taught in the above referenced application addresses these problems by defining an interface and a compensation mechanism to a set of transport services that provide connections across a plurality of transport protocols. By providing a transport boundary between the networks and the applications resident on systems, MPTN provides a common transport interface so that messages from the applications can be transported over any protocol in the network.

As shown in FIG. 1, hosts 20 and 22 are coupled to the multiprotocol transport network 16. While the MPTN 16 appears as though it is a single logical network having a single protocol, host 20 is coupled to a first network 10 which communicates via protocol X, e.g., TCP/IP, and host 22 is coupled to a second network 12 which communicates via protocol Y, e.g., NetBIOS.

Application program A 24 resident in one of the hosts 20 coupled to the MPTN network 16 wishes to communicate to application B 26 resident in another host 22 also coupled to the network 16. Upon application A's call to the socket layer 27, a socket is created by the socket layer 27 defining application A as an endpoint.

Sockets contain information about their type, the supporting protocols used in the transport layer 28 and their state. A connection request goes through the transport layer 28, the network layer 29 and the network interface layer 30 which add the necessary control and data information before the message is sent out on the network 54. Compensation for the differences between protocol y and X is carried out by the transport layer as taught by the above referenced application.

FIG. 2 depicts the code modules in memory at a computer system which is coupled to the multiprotocol transport network in greater detail. The socket programming interface 60 and common transport semantics 62 correspond to the socket layer and separate the applications 64, 66, 68 from the service drivers 70, 72, 74. Three types of applications 64, 66, 68 are shown in the application layer. To utilize the socket structure of the present invention, NetBIOS applications would be rewritten to the socket API to become the NetBIOS socket application 66. The standard local IPC socket application 64 and TCP/IP applications 68 are already written to a standardized socket programming interface and so would require an minimum of revisions. The common transport semantics 62, include the socket control blocks, which will be described in greater detail below. An interface between the socket layer and the transport layer is comprised by the local IPC service driver 70, the NetBIOS service driver 72 and the INet service driver 74 which correspond to the applications in the application layer. The service drivers are used to emulate the common transport semantics for the transport protocol drivers in the transport layer. In the present invention, they may also contain the compensating means described in the above referenced application which converts a message intended for a first protocol by the application to a second protocol supported by the network. The transport layer includes the NetBIOS 76 and TCP/IP 78 protocol drivers which cause the application message to conform to the protocol format. There is no corresponding local IPC module as it describes a local protocol whose messages are not placed on the network. The network and network interface layers are not pictured, the latter would include device drivers for the I/O hardware which couples the computer system to the network, e.g. a token ring or ethernet driver; the former might include drivers written to the well known NDIS interface.

A socket is the object in the Berkeley version of UNIX(TM) from which messages are sent and received. Sockets are created within a communication domain as files are created within a file system. A communication domain summarizes the sets of protocols, the naming conventions, the hardware which may be a particular network and may use an address which refers to the communication domain. The internet domain specified by the address family AF_INET; the NetBIOS domain is referenced by the address family AF_NETBIOS. A connection is a means by which the identity of the sending socket is not required to be sent with each packet of data. The identity of each communication endpoint is exchanged prior to any transmission of data and is maintained at the transmitting and receiving nodes, so that the distributed applications at each end can request the socket information at any time.

When an application creates a socket endpoint for a certain protocol and the protocol chosen by the transport network matches that protocol, native networking has occurred. For example, the INet protocol is used to support the INet address family and NetBIOS supports the NetBIOS address family. On the other hand, when the transport protocol does not match the socket endpoint of an application it is termed non-native networking. Using the present invention, however, the application is unaware that a different transport protocol is being used to connect with other nodes in the network.

In the present invention, the socket interface is used to connect between replicas of a distributed application or the client and server portions of a client/server application using a variety of transport protocols. The application can select the transport protocol or request that the socket layer determine the protocol. With the non-native networking feature of the present invention, applications written to communicate with one another using one protocol can choose to communicate on another transport protocol which might be optimized for the network environment. For example, an application written for TCP/IP could communicate using the NetBIOS protocol over the network,giving the distributed application a significant performance gain.

A conventional socket control block is depicted in FIG. 3, a socket control block 100 contains information about the socket, including send and receive data queues, their type, the supporting protocol, their state and socket identifier. The socket control block 100 includes a protocol switch table pointer 104 and a protocol control block pointer 106. The pointers are used to locate the protocol switch table (not pictured) and the protocol control block 102 respectively. The protocol switch table contains protocol related information, such as entry points to protocol, characteristics of the protocol, certain aspects of its operation which are pertinent to the operation of the socket and so forth. The socket layer interacts with a protocol in the transport layer through the protocol switch table. The user request routine PR_USRREQ is the interface between a protocol and the socket. The protocol control block contains protocol specific information which is used by the protocol and is dependent upon the protocol.

A socket control block according to the present invention is shown in FIG. 4. Here, the socket control block 110 is shown broken into two sections, the main socket control block which is largely identical to the conventional socket control block above and the MPTN extension 112 which contains many of the features necessary for the present invention. Both are linked together by a multiprotocol transport network extension 113. Each of the protocols which could be potentially used to send or receive data is sent a request to create a protocol control block 120, 122 at the time the new socket is created. After a selection procedure, the new socket then contains information regarding each selected protocol, the protocol switch table pointer 114, 118, the protocol control block pointer 116, 119, and a pointer to a interface address if applicable for the particular protocol (not pictured). As above, the protocol switch table pointer refers to a respective protocol switch table which defines various entry points for that protocol. Also, the protocol control blocks 120, 122 contain protocol specific information.

At the time of socket creation, no connection is made to any particular protocol to the application endpoint. Connection implies an association of the connection-requestor socket of the requesting application to another connection-acceptor socket. It can be viewed as a communication wire running between the two sockets that are "connected", potentially in two different continents, providing the ability for both of them to send and receive messages. There is no difference between a transmitting socket and a receiving socket for the present invention. After a connection, each can send or receive data.

The decision on which protocol to use is delayed until the time that the connection is actually made. At the time of establishing a connection, any of the network interfaces in a protocol could be used. The order in which the pointers which refer to the protocols and network interfaces is based on user preference, information from the named service unit or the capability of the protocol the socket extension 112 which contains the pointers 114, 116, 118 and 119, also includes state information on the socket and the multiprotocol transport network. The socket extension 112 is used by the socket layer to manage the socket and MPTN states. From the list of available protocols, the socket layer picks those protocols that support the requested socket domain and the type of communication (datagrams, streams, etc), as described in the flow diagram for socket creation described below.

A sample socket control block structure is given in the code below:

```
/* socketva.h.....
 * Kernel structure per socket.
 * Contains send and receive buffer queues.
 * handle on protocol and pointer to protocol
 * private data, error information and MPTN extensions.
 * The first part of this structure ( upto the so_mptn field ) is    identical
 * to the BSD4.3 socket control block.
 */
struct socket {
     short   so_type;                 /* generic type, see socket.h */
     short   so_options;              /* from socket call, see socket.h */
     short   so_linger;               /* time to linger while closing */
     short   so_state;                /* internal state flags SS_*, below */
     caddr_t so_pcb;                  /* protocol control block */
     struct  protosw far *so_proto;      /* protocol handle */
/*
 * Variables for connection queueing.
 * Socket where accepts occur is so_head in all subsidiary sockets.
 * If so_head is 0, socket is not related to an accept.
 * For head socket so_q0 queues partially completed connections,
 * while so_q is a queue of connections ready to be accepted.
 * If a connection is aborted and it has so_head set, then
 * it has to be pulled out of either so_q0 or so_q.
 * We allow connections to queue up based on current queue lengths
 * and limit on number of queued connections for this socket.
 */
     struct  socket far *so_head;     /* back pointer to accept socket */
     struct  socket far *so_q0;       /* queue of partial connections */
     struct  socket far *so_q:           /* queue of incoming connections */
     short   so_q0len:                /* partials on so_q0 */
     short   so_qlen;                 /* number of connections on so_q */
     short   so_qlimit;               /* max number queued connections */
     short   so_timeo:                /* connection timeout */
     u_short so_error;                /* error affecting connection */
     short   so_pgrp;                 /* pgrp for signals */
     u_long  so_oobmark;              /* chars to oob mark */
/ * Variables for socket buffering.
 */
        struct  sockbuf {
                u_long  sb_cc;        /* actual chars in buffer */
                u_long  sb_hiwat;     /* max actual char count */
                u_long  sb_mbcnt;     /* chars of mbufs used */
                u_long  sb_mbmax;     /* max chars of mbufs to use */
                u_long  sb_lowat;     /* low water mark (not used yet) */
                struct  mbuf far *sb_mb;    /* the mbuf chain */
                struct  proc far *sb_sel;    /* process selecting read/write */
                short   sb_timeo;     /* timeout (not used yet) */
                short   sb_flags:     /* flags, see below */

      } so_rcv, so_snd;

    /* MPTN SOCKET EXTENSION */

       struct m_esock far * so_mptn;    /* socket MPTN extensions */

#define SB_MAX    ((u_long)(64*1024L))       /* max chars in sockbuf */
#define SB_LOCK          0x01         /* lock on data queue (so_rcv only) */
#define SB_WANT          0x02         /* someone is waiting to lock */
#define SB_WAIT          0x04         /* someone is waiting for data/space */
#define SB_SEL           0x08         /* buffer is selected */
#define SB_COLL          0x10         /* collision selecting */
};

  /* Socket extensions for MPTN
   * The socket control block points to this structure which contains
   * all the MPTN related socket extensions.
   * Since m_esock, the MPTN extensions to sockets, is contained in one
   * mbuf, we could use the rest of mbuf space to accommodate the pcb
   * pointers.
```

```
*/

/* defines the structure for storing additional pointers.
 * used in m_esock.
 * The structure m_addr defines the address format.It is defined in mptndef.h.
 * The structure m_info defines the user specified connection characteristics
 * and is defined in mptndef.h.
 * The structure bnlst defines the user configured protocol preference list
 * and is defined in mptndef.h.
 *
 */
struct m_sopcb {
        struct protosw far * so_proto:   /* protocol switch ptr */
        caddr_t so_pcb;                  /* pointer to the PCB */
        struct m_addr far * so_bnsap:    /* the network interface             that
                                          * the PCB refers to */
          };

struct m_conn_stat {
    char  type;              /* DST_BNSAP_FOUND,USE_PREF_LIST,CACHE,
                              * GW_NEXT_HOP, GW_BNSAP */
    char  index:         /* current network..as an index in to the
                          * pref. list as the case is.
                          */
    caddr_t  ftnam:         /* destination name for ABM case, Gateway
                             * cases, it is dst_b_nam.
                             * In the native case or cache case,contains
                             * the dest A-addr
                             */

    struct bnlst far *prflst; /* pointer to the pref list */
             };

  struct m_esock {
   int (* so_upcall ) ():          /* user upcall address */
   struct socket far * so_relay; /* relay socket pointer:for gateway only */
     struct m_info far * so_info;   /* connection chars given at m_create()*/
     struct m_info far * so_cinfo:  /* conn.char of a connection */
     short so_domain:               /* user specified domain */
     struct m_addr far * so_lanam;  /* local user name */
     struct m_addr far * so_panam;  /* peer  user name */
     struct mbuf far * so_ctdata;   /* connection/termination data */
     struct mbuf far * so_expdat;   /* expedited data */
     struct socket far * so_next;   /* linked list of nonnative scbs.to search
                                     * for user-names*/
     unsigned so_mptn_flag;         /* mptn flags having the following defs*/
     struct m_conn_stat conn_stat;  /* to maintain mptn_connect status */
     short so_pcbnum ;              /* number of pcbs currently in sopcb[]  */
     long  so_pcbstat:              /* pcb state:bit position corresponds to
                                     * pcb array; 0->in-use; else not-in-use*/
     struct  m_sopcb sopcb [MPTN_MAXPCB];/* array of m_sopcb structures */

#define MPTN_NONNATIVE         0x01     /* indicates nonnative connection */
#define MPTN_NATIVE            0x02     /* indicates native connection */
#define MPTN_SO_BIND           0x04     /* sock mptn state=>addr bound */
#define MPTN_SO_UNBIND         0x08     /* sock mptn state=>addr not bound*/
#define MPTN_SO_CONN           0x10     /* sock mptn state=>connect req is made*/
#define MPTN_SO_NOMORE_CONN    0x20     /* =>there will be no more conn. re-try
                                         * over other networks/protocols*/
#define MPTN_SO_LISTEN         0x40     /* init state of a passive sock */
#define MPTN_SO_CON_HEAD_WAIT 0x80      /* native con set up..waiting for MPTN
                                         * connect header  */
#define MPTN_SO_CON_HEAD_RCVD 0x0100    /* passive node...mptn con rcvd..*/
#define MPTN_SO_CON_HEAD_SENT 0x0200    /* Active node....mptn con sent ..*/
#define MPTN_SO_CON_ESTABLISHED 0x0400 /* Connection established ...*/
#define MPTN_SO_CLOSED         0x0800   /* local socket close issued */
#define MPTN_SO_BCAST_RCV      0x1000   /* the socket is enabled to rcv
                                         * broadcast dgms.*/
#define MPTN_SO_IN_ADDR_ANY    0x2000   /* the socket is bound to in_addr_any */

  };

/*
```

```
 * Socket state bits:
 */
#define SS_NOFDREF              0x001   /* no file table ref any more */
#define SS_ISCONNECTED          0x002   /* socket connected to a peer */
#define SS_ISCONNECTING         0x004   /* in process of connecting to peer */
#define SS_ISDISCONNECTING      0x008   /* in process of disconnecting */
 #define SS_CANTSENDMORE         0x010   /* can't send more data to peer */
#define SS_CANTRCVMORE          0x020   /* can't receive more data from peer */
#define SS_RCVATMARK            0x040   /* at mark on input */

#define SS_PRIV                 0x080   /* privileged for broadcast, raw... */
#define SS_NBIO                 0x100   /* non-blocking ops */
#define SS_ASYNC                0x200   /* async i/o notify */

#define SS_CANCEL               0x4000  /* cancel call */
#define SS_PUSHSEEN             0x8000  /* seen push   */

/* Rest of the info is the same as in the BSD4.3 socketva.h */
Copyright, IBM Corp. 1993
```

Conventional socket creation starts with a call to the socket API. A domain table is searched for the address family, the type and protocol which is desired by the application. If a match is found the protocol switch table entry is set in the socket control block. Next, the user requests entry and the selected protocol is called to create of the protocol control block. If a match is not found in the domain table for the address family, type and protocol desired by the application, an error is returned to the application.

FIG. 5 depicts the process of creating a socket according to the present invention. The process begins with a socket creation request 150 from an application to the socket API, the application wishes to send or receive data across the network. The command to the socket API for the request takes the form of socket=(AF, *, type, proto). "AF" refers to the address family and communication domain; "type" refers to one of the socket types defined in a system header file. The "proto" or protocol parameter is used to indicate that a specific protocol is to be used. A test is performed in step 152 to determine if "proto" is specified. If so, the normal socket creation process in step 154 is carried out and the process ends, step 155.

If "proto" is not specified, the process continues to create a multiprotocol socket. Each protocol is associated with a protocol switch table. For each protocol switch table, step 155, tests are performed whether the type and address family of the requesting endpoint, steps 158 and 160, are matched by the protocol. In step 158 the test for "type" match is performed. If the test fails, the process returns for the next protocol switch table, step 156. If the test succeeds, the test for address family match is performed in step 160. Both the "type" and "family" are represented as integers. By matching, the socket layer compares the 'type' and "address family" field supplied by the user and the 'type' and "address family" fields in the protocol. If a match is found for both type and address family, the protocol is selected as a candidate protocol and set in the socket extension with pointers to the protocol switch table and protocol control block, step 162. If there is no match, the process determines whether the address family is supported non-natively in the network, step 164. If the protocol is supported, step 166, the protocol is selected as a candidate protocol and set in the socket extension with pointers to the protocol switch table and protocol control block, step 162. If the protocol is not supported, in step 166, a test is performed to determine whether it is the last protocol switch table. If not, the process returns to step 156. If it is the last protocol switch table, the process ends, step 170. The protocol pointers can be reordered within the extension according to the application or user preference through the use of a configuration tool or according to information from the named server.

In FIG. 6, the process to establish a connection using the multiprotocol socket is described. The process begins in step 200 where the socket layer tries connecting to a specified destination using the first protocol from the list of protocols in the socket extension. The choice of which of the protocols to try first can be based on the configuration of user specified preference order of protocols. If the connection succeeds, step 222, the process ends in step 223. If not, the next protocol is used to try to establish a connection in step 224. Tests are performed determine whether a connection is made, step 226. If so, the process ends. If a connection is not made, a test is performed to determine whether it is the last protocol in the socket extension. If not, the next protocol in the extension is tested, step 224. If all the protocols used when creating the socket have failed to provide the connection, the transport provider returns a notification of failure to the application, step 232.

As mentioned previously, the invention finds use in a plurality of computers which are part of a network such as a Local Area Network or wide Area Network. Although the specific choice of computer in these networks is limited only by performance and storage requirements, computers in the IBM PS/2 (TM) series of computers could be used in the present invention. For additional information on IBM's PS/2 series of computers, the reader is referred to Technical Reference Manual Personal Systems/2 Model 50, 60 Systems IBM Corporation, Part No. 68X2224 Order Number 568X-2224 and Technical Reference Manual Personal Systems/2 (Model 80) IBM Corporation Part No. 68X 2256 Order Number S68X-2254. One operating system which an IBM PS/2 personal computer may run is IBM's OS/2 2.0 (TM). For more information on the IBM OS/2 2.0 Operating System, the reader is referred to OS/2 2.0 Technical Library. Programming guide Vol. 1. 2. 3 Version 2.00 Order Nos. 10G6261, 10G6495, 10G6494. In the alternative, the computer system might be in the IBM RISC System/6000 (TM) line of computers which run on the AIX (TM) operating system. The various models of the RISC System/6000 are described in many publications of the IBM Corporation, for example, RISC System/6000. 7073 and 7016 POWERstation and POWERserver Hardware Technical reference, Order No. SA23-2644-00. The AIX operating system is described in General Concepts and Procedure--AIX Version 3 for RISC System/6000 Order No. SC23-2202-00 as well as other publications of the IBM Corporation. In lieu of the cited references, the reader is offered the following general description of a computer system which could be utilized to practice the present invention.

In FIG. 7, a computer 310, comprising a system unit 311, a keyboard 312, a mouse 313 and a display 314 are depicted. The screen 316 of display device 314 is used to present the visual changes to the data object. The graphical user interface supported by the operating system allows the user to use a point and shoot method of input by moving the pointer 315 to an icon representing a data object at a particular location on the screen 316 and press one of the mouse buttons to perform a user command or selection.

FIG. 8 shows a block diagram of the components of the computer shown in FIG. 7. The system unit 11 includes a system bus or plurality of system buses 21 to which various components are coupled and by which communication between the various components is accomplished. The microprocessor 322 is connected to the system bus 321 and is supported by read only memory (ROM) 323 and random access memory (RAM) 324 also connected to system bus 321. A microprocessor in the IBM multimedia PS/2 series of computers is one of the Intel family of microprocessors including the 386 or 486 microprocessors. However, other microprocessors included, but not limited to, Motorola's family of microprocessors such as the 68000, 68020 or the 68030 microprocessors and various Reduced Instruction Set Computer (RISC) microprocessors manufactured by IBM, Hewlett Packard, Sun, Intel, Motorola and others may be used in the specific computer.

The ROM 323 contains among other code the Basic Input-Output system (BIOS) which controls basic hardware operations such as the interaction and the disk drives and the keyboard. The RAM 324 is the main memory into which the operating system, transport providers and application programs are loaded. The memory management chip 325 is connected to the system bus 321 and controls direct memory access operations including, passing data between the RAM 324 and hard disk drive 326 and floppy disk drive 327. The CD ROM 332 is also coupled to the system bus 321.

Also connected to the system bus 321 are various I/O controllers: The keyboard controller 328, the mouse controller 329, the video controller 330, and the audio controller 331 which control their respective hardware, the keyboard 312, the mouse 313, the display 314 and the speaker 315. Also coupled to the system bus 321 is the digital signal processor 333 which corrects the sound produced by the speaker 315 and is preferably incorporated into the audio controller 331. An I/O controller 340 such as a Token Ring Adapter enables communication over a network 342 to other similarly configured data processing systems.

While the invention has been described with respect to particular embodiments above, it will be understood by those skilled in the art that modifications may be made without departing from the spirit and scope of the present invention. The preceding description has described the principles of the present invention in terms of a particular communications endpoint object, a socket, in the socket layer between the application layer and transport layer. The principles of the invention could be extended to provide similarly configured communication endpoint objects in other layers. For example, an object in the network interface layer could be used to monitor a plurality of underlaying MAC drivers to that data could be sent or received over a plurality of MAC protocols to different types of networks. These embodiments are for purposes of example and illustration only and are not to be taken to limit the scope of the invention narrower than the scope of the appended claims.

Claims

1. A method for communicating between nodes in a computer network in which a plurality of protocols are useable by network nodes, the method comprising the steps of:

creating communication endpoint objects defining communication parameters for respective network nodes, the objects having information about the plurality of protocols which are useable by said node for inter-node communication; and

at the time communication is requested between said nodes, establishing a connection between said nodes using a selected one of the plurality of protocols.

2. A method according to claim 1, wherein the selected protocol is chosen based on user preferences.

3. A method according to claim 1, wherein the selected protocol is chosen based on a capability of the selected protocol.

4. A method according to any one of the preceding claims, wherein the creating step comprises the steps of:

requesting information of each of the plurality of protocols;

selecting a set of protocols from the plurality of protocols, which selected protocols are useable by the respective nodes;

building a protocol control block for each of the selected protocols; and,

inserting a pointer in the communication endpoint object for each protocol control block of a selected protocol.

5. A method according to claim 4, wherein the order in which the pointers are inserted is based on user preference.

6. A method according to claim 4 wherein the selecting step comprises the steps of:

determining whether there is a match for a first protocol for a first set of protocol parameters, the first protocol corresponding to a first pointer in the socket; and,

responsive to finding no match for the first protocol, determining whether the first protocol is supported non-natively in the network.

7. A system for communicating between nodes in a computer network in which a plurality of protocols are useable by network nodes, the system comprising:

means for creating communication endpoint objects defining communication parameters for respective network nodes, which objects have information about the plurality of protocols useable by respective computer systems each coupled to one or more nodes in the network; and,

means for establishing a connection between a first and a second computer system using a selected one of the plurality of protocols at the time communication is requested between nodes on the different systems.

8. A system according to claim 7, which further comprises:

means for selecting a set of protocols from said plurality of protocols;

means for requesting information from each of the plurality of protocols;

means for building a protocol control block for each of the selected protocols; and,

means for inserting a pointer in the communication endpoint object for each protocol control block for a selected protocol.

9. A system according to claim 8, which further comprises:

means for determining whether there is a match for a first protocol for a first set of protocol parameters, the first protocol corresponding to a first pointer in the socket; and,

means for determining, responsive to finding no match for the first protocol, whether the first protocol is supported non-natively in the network.

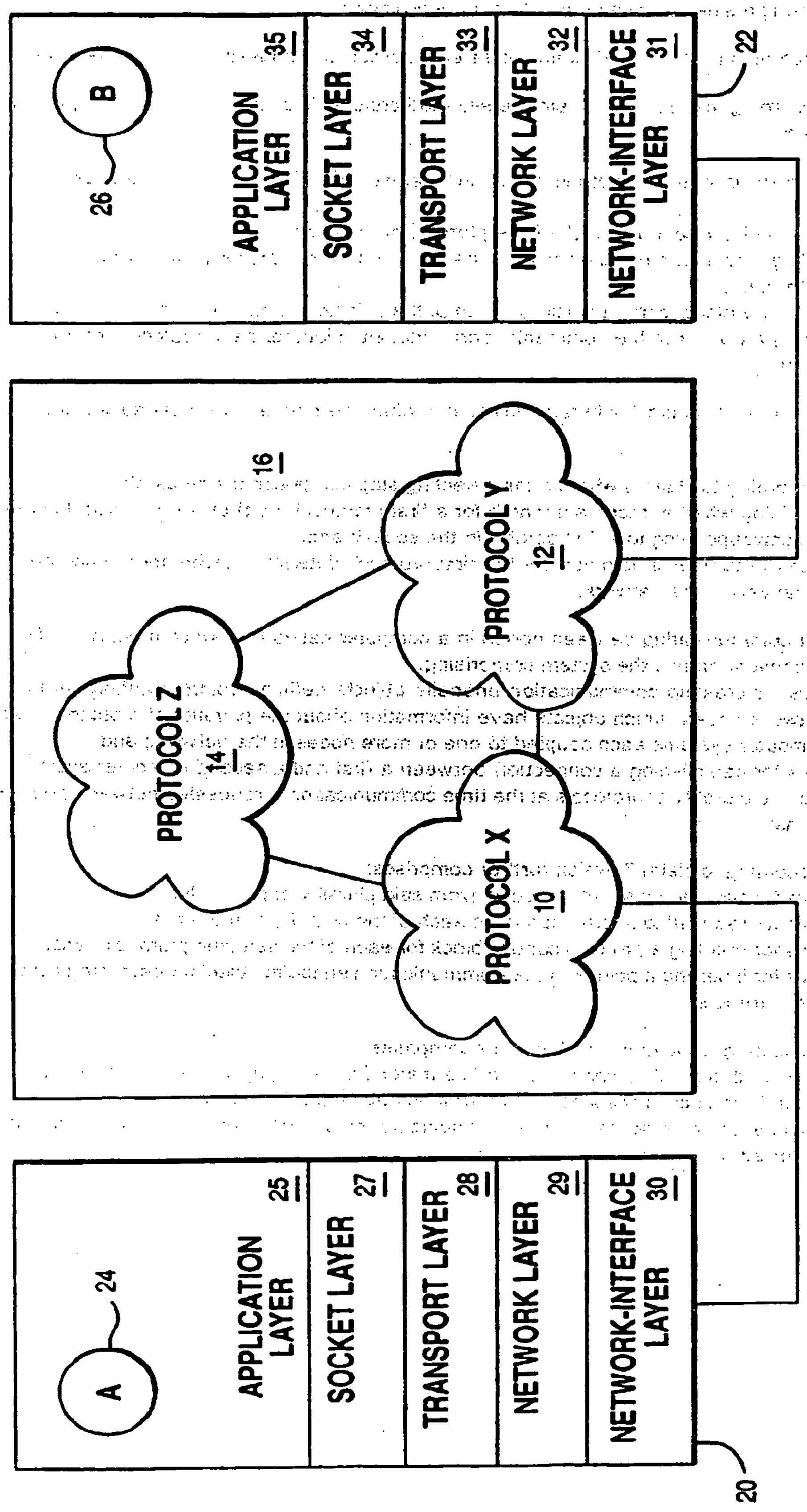

B
26
APPLICATION LAYER
35
SOCKET LAYER
34
TRANSPORT LAYER
33
NETWORK LAYER
32
NETWORK-INTERFACE LAYER
31
22
16
PROTOCOL Y
12
PROTOCOL Z
14
PROTOCOL X
10
A
24
APPLICATION LAYER
25
SOCKET LAYER
27
TRANSPORT LAYER
28
NETWORK LAYER
29
NETWORK-INTERFACE LAYER
30
20

FIG. 1

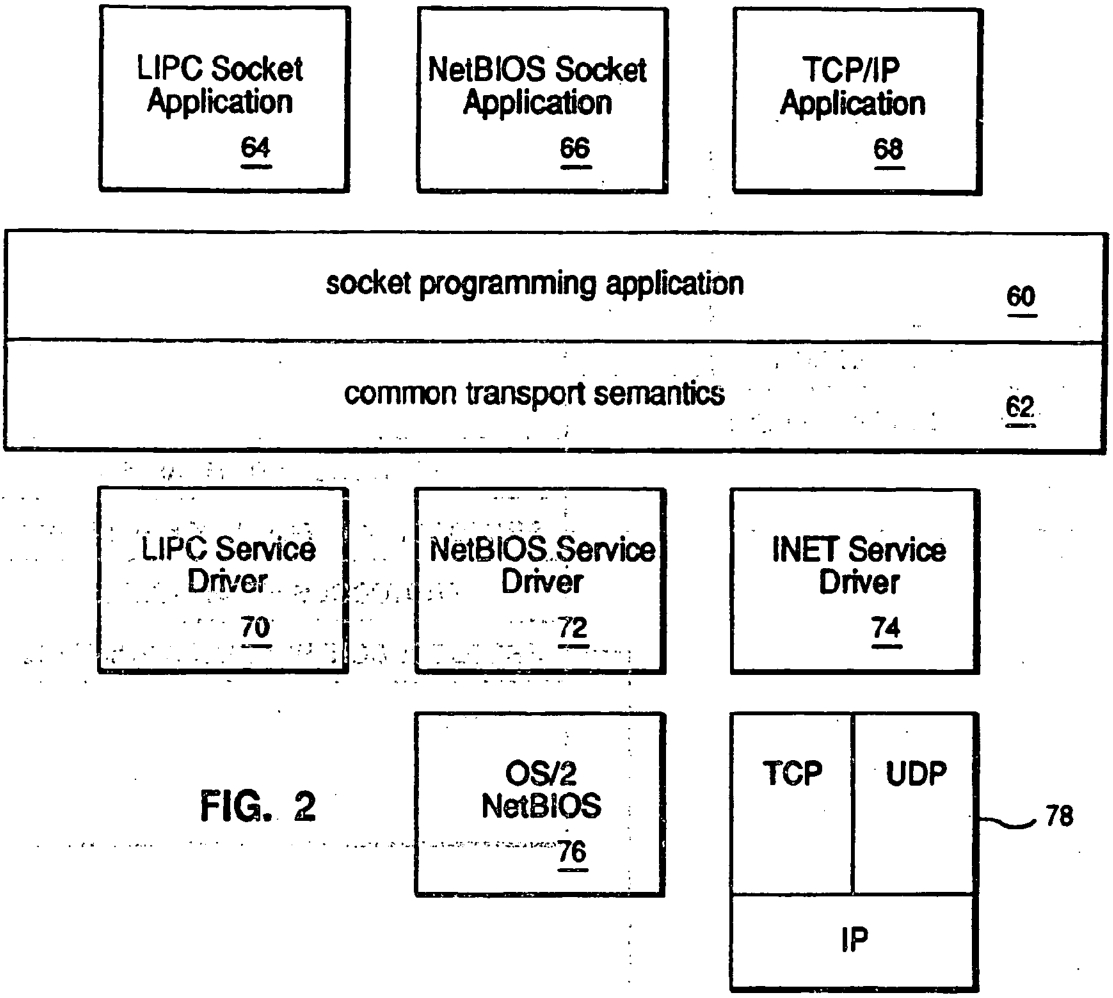


FIG. 2

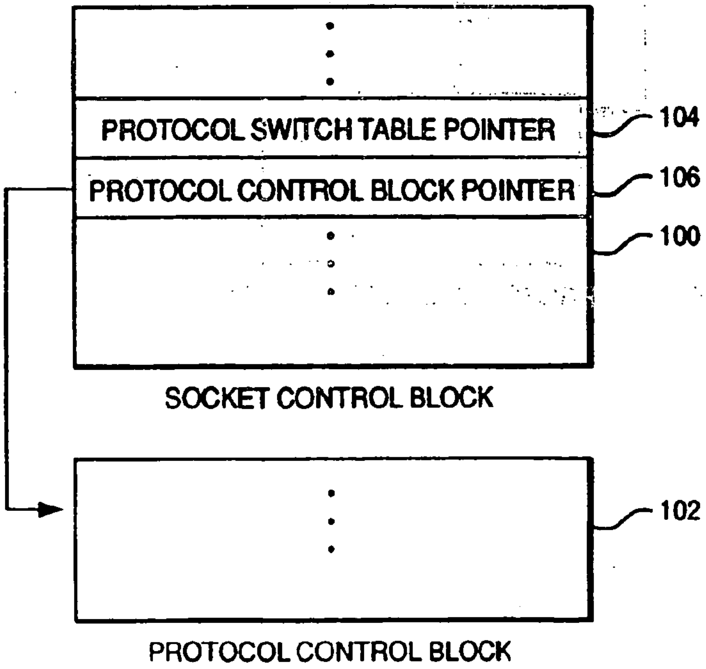


FIG. 3
PRIOR ART

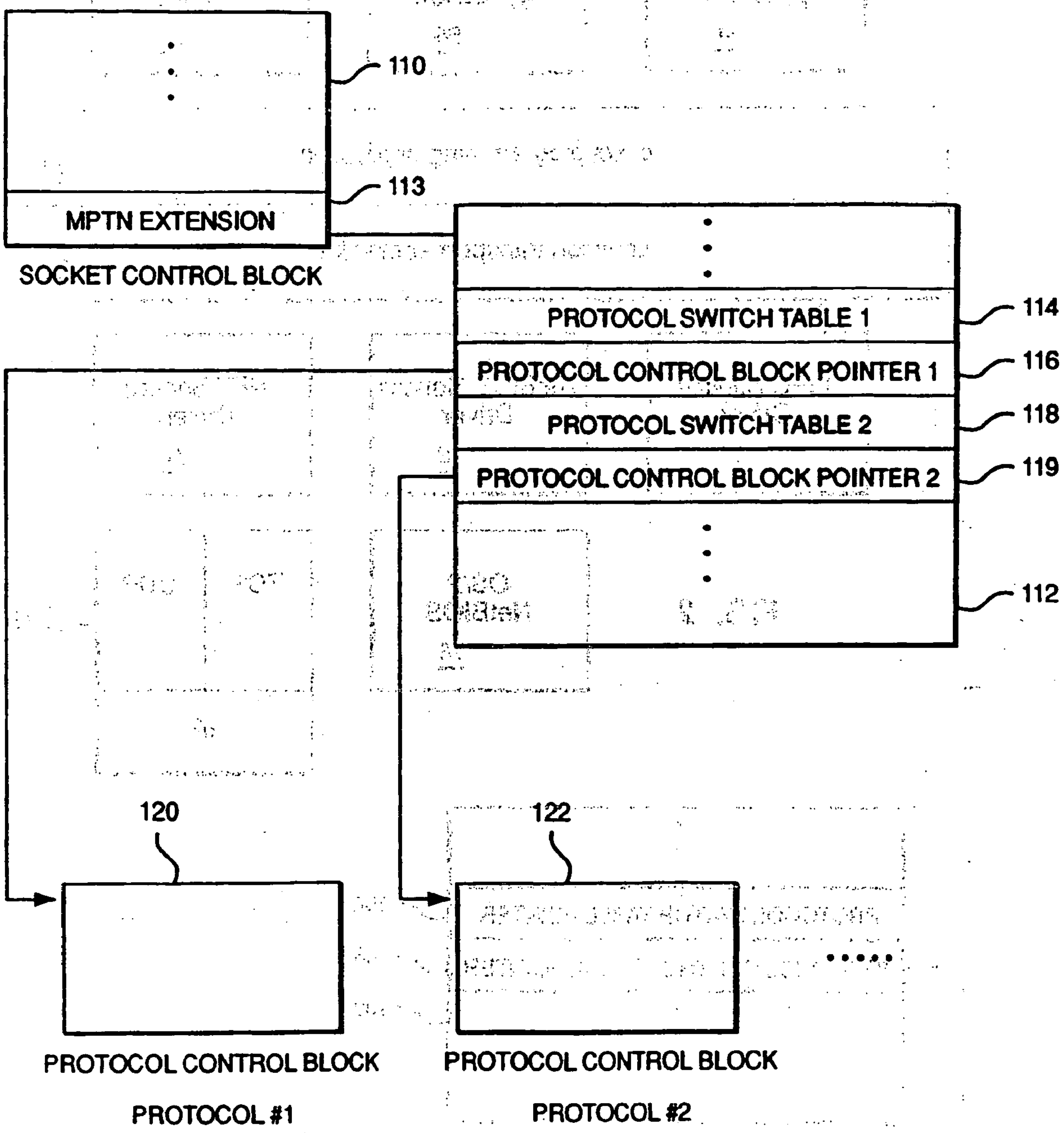

110
113
MPTN EXTENSION
SOCKET CONTROL BLOCK
PROTOCOL SWITCH TABLE 1
114
PROTOCOL CONTROL BLOCK POINTER 1
116
PROTOCOL SWITCH TABLE 2
118
PROTOCOL CONTROL BLOCK POINTER 2
119
112
120
122
PROTOCOL CONTROL BLOCK
PROTOCOL #1
PROTOCOL CONTROL BLOCK
PROTOCOL #2

FIG. 4

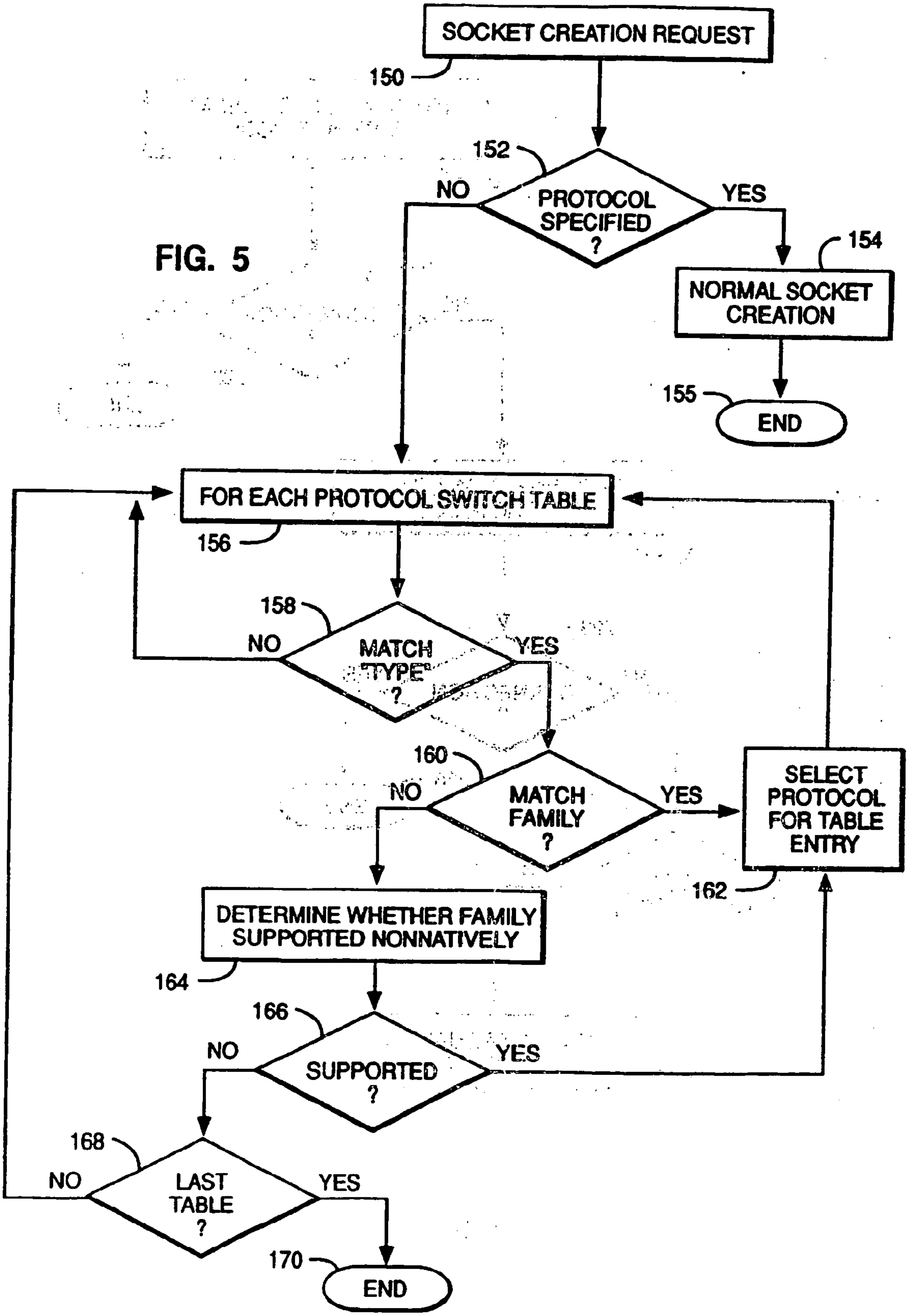

SOCKET CREATION REQUEST
150
152
PROTOCOL SPECIFIED ?
NO
YES
154
NORMAL SOCKET CREATION
155
END
FOR EACH PROTOCOL SWITCH TABLE
156
158
MATCH "TYPE" ?
NO
YES
160
MATCH FAMILY ?
NO
YES
SELECT PROTOCOL FOR TABLE ENTRY
162
DETERMINE WHETHER FAMILY SUPPORTED NONNATIVELY
164
166
SUPPORTED ?
NO
YES
168
LAST TABLE ?
NO
YES
170
END

FIG. 5

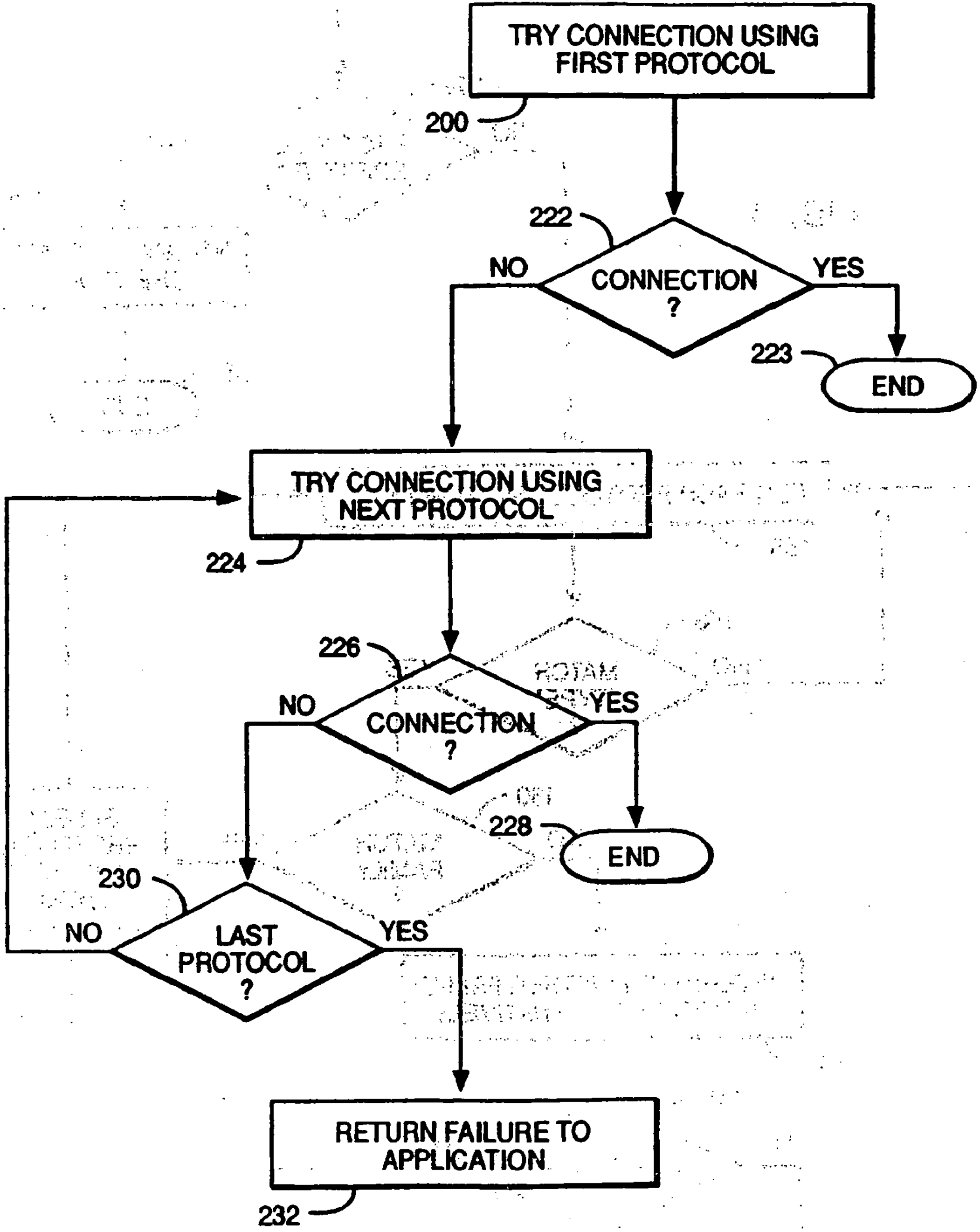

TRY CONNECTION USING FIRST PROTOCOL
200
222
CONNECTION ?
NO
YES
223
END
TRY CONNECTION USING NEXT PROTOCOL
224
226
CONNECTION ?
NO
YES
228
END
230
LAST PROTOCOL ?
NO
YES
RETURN FAILURE TO APPLICATION
232

FIG. 6

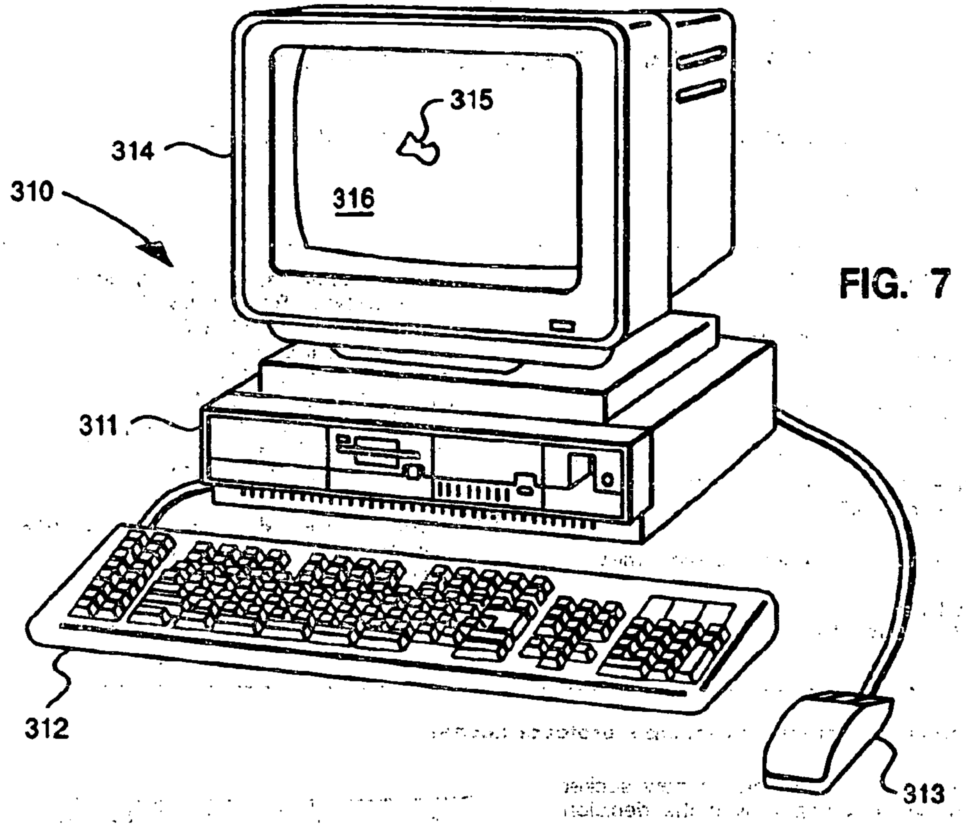

310
314
315
316
311
312
313

FIG. 7

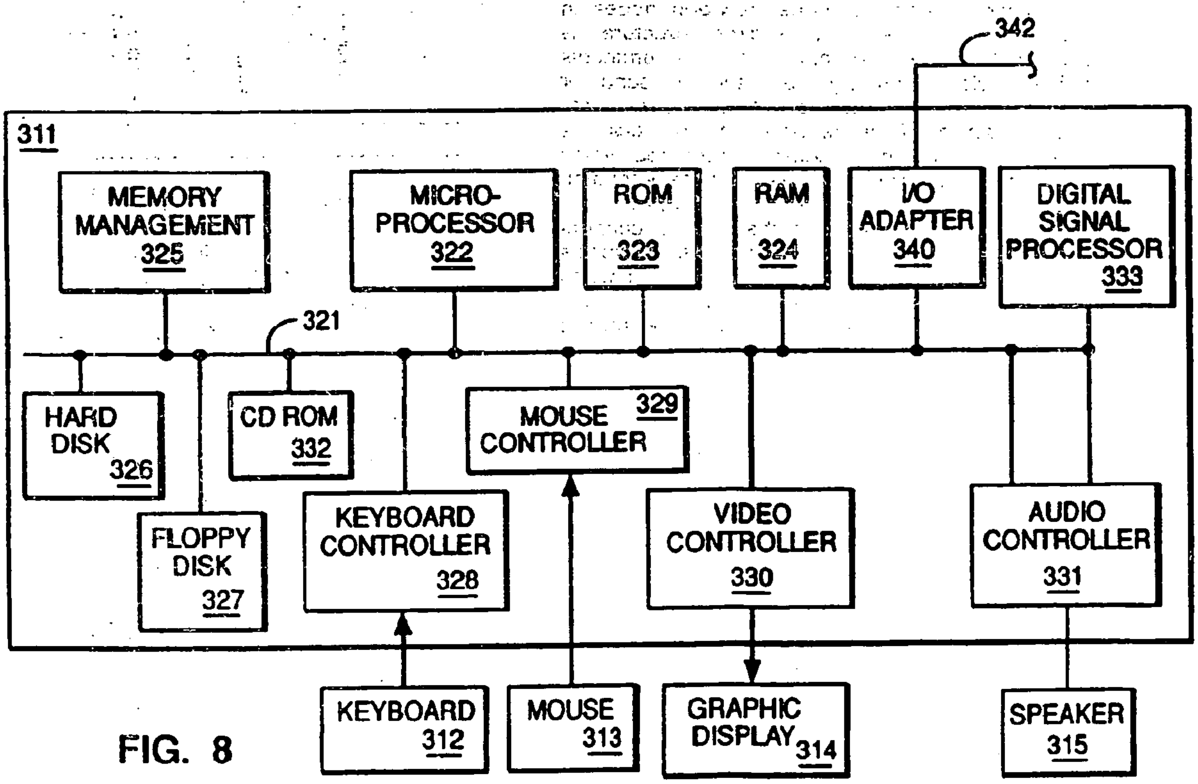

342
311
MEMORY MANAGEMENT 325
MICRO-PROCESSOR 322
ROM 323
RAM 324
I/O ADAPTER 340
DIGITAL SIGNAL PROCESSOR 333
321
HARD DISK 326
CD ROM 332
MOUSE CONTROLLER 329
FLOPPY DISK 327
KEYBOARD CONTROLLER 328
VIDEO CONTROLLER 330
AUDIO CONTROLLER 331
KEYBOARD 312
MOUSE 313
GRAPHIC DISPLAY 314
SPEAKER 315

FIG. 8

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 613 274 A3**

(12) **EUROPEAN PATENT APPLICATION**

(21) Application number : **94300677.5**

(22) Date of filing : **28.01.94**

(51) Int. Cl.^{5} : **H04L 29/06**

(30) Priority : **29.01.93 US 15366**

(43) Date of publication of application :
31.08.94 Bulletin 94/35

(84) Designated Contracting States :
DE FR GB

(88) Date of deferred publication of search report :
18.01.95 Bulletin 95/03

(71) Applicant : **International Business Machines Corporation
Old Orchard Road
Armonk, N.Y. 10504 (US)**

(72) Inventor : **Sharma, Mohan Byrappa
12148 Jollyvill Road No.316
Austin, Texas 78759 (US)**
Inventor : **Yeung, Yue
11714 D-K Ranch Road
Austin, Texas 78759 (US)**
Inventor : **Cheng, Chungsiang
10629 Floral Drive
Austin, Texas 78759 (US)**

(74) Representative : **Moss, Robert Douglas
IBM United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester Hampshire SO21 2JN (GB)**

(54) **Socket structure for concurrent multiple protocol access.**

(57) In a multiprotocol environment, a new socket structure is provided which moves the decision on which protocol to use until the time that the connection is actually made between nodes in the network. The new socket structure is created for every endpoint. All the protocols which could potentially be used to send or receive data is sent a request to create a protocol control block at the time the new socket is created. A selection process determines which of the protocols could be used by the endpoint. The new socket then contains information on each selected protocol. At the time a connection is established, any of the selected protocols could be used. The choice of which protocol to use can be based on user preferences, which protocols are available, the name of the service unit.

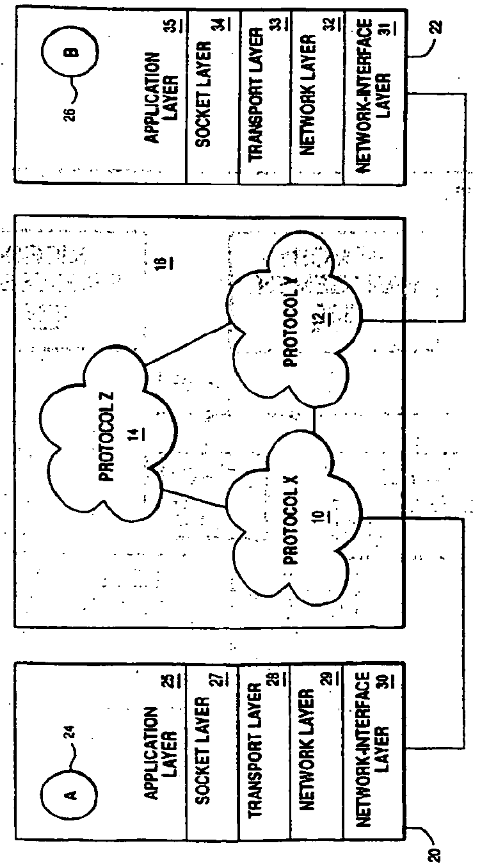


FIG. 1

Jouve, 18, rue Saint-Denis, 75001 PARIS

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 94 30 0677

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.5)
Y	COMPUTER COMMUNICATIONS., vol.10, no.1, February 1987, GUILDFORD GB pages 21 - 29 D.COFFIELD ET AL 'TUTORIAL GUIDE TO UNIX SOCKETS FOR NETWORK COMMUNICATIONS' * page 22, left column, line 35 - page 25, left column, line 7 * ---	1,7	H04L29/06
Y	IEEE TRANSACTIONS ON SOFTWARE ENGINEERING., vol.17, no.1, January 1991, NEW YORK US pages 64 - 76 N.C.HUTCHINSON ET AL 'THE X-KERNEL: AN ARCHITECTURE FOR IMPLEMENTING NETWORK PROTOCOLS' * paragraph II * ---	1,7	
A	IBM SYSTEMS JOURNAL., vol.27, no.2, 1988, ARMONK, NEW YORK US pages 90 - 104 M.S.KOGAN ET AL 'THE DESIGN OF OPERATING SYSTEM/2' * page 99, left column, line 17 - page 100, right column, line 35 * -----	1-9	TECHNICAL FIELDS SEARCHED (Int.Cl.5) H04L

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	15 November 1994	Canosa Areste, C

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

Blank (uspto)

Network Working Group
Internet-Draft
Expires: May 16, 2001

I. Cooper
Equinix
P. Gauthier
Inktomi Corporation
J. Cohen
(Microsoft Corporation)
M. Dunsmuir
(RealNetworks, Inc.)
C. Perkins
Sun Microsystems, Inc.
November 15, 2000

Web Proxy Auto-Discovery Protocol
draft-cooper-webi-wpad-00.txt

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on May 16, 2001.

Copyright Notice

Abstract

A mechanism is needed to permit web clients to locate nearby (caching) web proxy. Current best practice is for end users to hand configure their web client (i.e., browser) with the URL of an "auto configuration file". In large environments this presents a formidable support problem. It would be much more manageable for

1. Prior Work

This memo is built on the prior work of Paul Gauthier, Josh Cohen, Martin Dunsmuir and Charles Perkins. Their efforts in producing previous versions of this work are acknowledged with thanks.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP4[7].

3. Introduction

The problem of locating nearby web proxies cannot wait for the implementation and large scale deployment of various upcoming resource discovery protocols. The widespread success of the HTTP protocol and the recent popularity of streaming media has placed unanticipated strains on the networks of corporations, ISPs and backbone providers. There currently is no effective method for these organizations to realize the obvious benefits of web caching without tedious and error-prone configuration by each and every end user.

The de-facto mechanism for specifying a web proxy configuration in web clients is the download of a script or configuration file named by a URL. Users are currently expected to hand configure this URL into their browser or other web client. This mechanism suffers from a number of drawbacks:

- o Difficulty in supporting a large body of end-users. Many users misconfigure their proxy settings and are unable to diagnose the cause of their problems.
- o Lack of support for mobile clients who require a different proxy as their point of access changes.
- o Lack of support for complex proxy environments where there may exist a number of proxies with different affinities for different clients (based on network proximity, for example). Currently, clients would have to "know" which proxy server was optimal for their use.

Currently available methods for resource discovery need to be exploited in the context of a well defined framework. Simple, functional and efficient mechanisms stand a good chance of solving this pressing and basic need. As new resource discovery mechanisms mature they can be folded into this framework with little difficulty.

This document is a specification for implementers of web client software. It defines a protocol for automatically configuring those clients to use a local proxy. It also defines how an administrator should configure various resource discovery services in their

Table of Contents

The client MUST assume port 80 if the successful discovery mechanism does not provide a port component.

PATH

The client MUST assume a path of "/wpad.dat" if the successful discovery mechanism does not provide a path component.

5. The Discovery Process

5.1 WPAD Overview

This sub-section will present a descriptive overview of the WPAD protocol. It is intended to introduce the concepts and flow of the protocol. The remaining sub-sections (Section 5.2-Section 5.7) will provide the rigorous specification of the protocol details. WPAD uses a collection of pre-existing Internet resource discovery mechanisms to perform web proxy auto-discovery. Readers may wish to refer to [1] for a similar approach to resource discovery, since it was a basis for this strategy. The WPAD protocol specifies the following:

- o how to use each mechanism for the specific purpose of web proxy auto-discovery
- o the order in which the mechanisms should be performed

The resource discovery mechanisms utilized by WPAD are as follows.

- o Dynamic Host Configuration Protocol (DHCP [3][4])
- o Service Location Protocol (SLP [5])
- o "Well Known Aliases" using DNS A records [6][9]
- o DNS SRV records [2][9]
- o "service: URLs" in DNS TXT records [11]

Of all these mechanisms only the DHCP and "Well Known Aliases" are required in WPAD clients. This decision is based on three reasons: these facilities are currently widely deployed in existing vendor hardware and software; they represent functionality that should cover most real world environments; they are relatively simple to implement.

DNS servers supporting A records are clearly the most widely deployed of the services outlined above. It is reasonable to expect API support inside most web client development environments (POSIX C, Java, etc). The hierarchical nature of DNS makes it possible to support hierarchies of proxy servers,

DNS is not suitable in every environment, unfortunately. Administrators often choose a DNS domain name hierarchy that does not correlate to network topologies, but rather with some organizational model (for example, foo.development.bar.com and foo.marketing.bar.com). DHCP servers, on the other hand, are frequently deployed with concern for network topologies. DHCP network to support WPAD compatible web clients.

While it does contain suggestions for web proxy software implementers, it does not make any specific demands of those parties.

4. Defining Web Proxy Auto-Discovery

As mentioned above, web client software currently needs to be configured with the URL of a proxy auto-configuration file or script. The contents of this script are vendor specific and not currently standardized. This document does not attempt to discuss the contents of these files (see[8] for an example file format).

Thus, the Web Proxy Auto-Discovery (WPAD) problem reduces to providing the web client a mechanism for discovering the URL of the Configuration File. Once this Configuration URL (CURL) is known, the client software already contains mechanisms for retrieving and interpreting the Configuration File (CFILE) to enable access to the specified proxy or proxies.

It is worth carefully noting that the goal of the WPAD process is to discover the correct CURL at which to retrieve the CFILE. The client is *not* trying to directly discover the name of the proxy. That would circumvent the additional capabilities provided by proxy Configuration Files (such as load balancing, request routing to an array of servers, automated fail-over to backup proxy [10][8]).

It is worth noting that different clients requesting the CURL may receive completely different CFILEs in response. The web server may send back different CFILES based on a number of criteria such as the "User-Agent" header, "Accept" headers, client IP address/subnet, etc. The same client could conceivably receive a different CFILE on successive retrievals (as a method of round-robin load balancing, for example).

This document will discuss a range of mechanisms for discovering the Configuration URL. The client will attempt them in a predefined order, until one succeeds. Existing widely deployed facilities may not provide enough expressiveness to specify a complete URL. As such, we will define default values for portions of the CURL which may not be expressible by some discovery mechanisms:

```
http://<HOST>:<PORT><PATH>
```

HOST
There is no default for this portion. Any succeeding discovery mechanism will provide a value for the <HOST> portion of the CURL. The client MUST NOT provide a default.

PORT

5.2 When to Execute WPAD

Web clients need to perform the WPAD protocol periodically to maintain correct proxy settings. This should occur on a regular basis corresponding to initialization of the client software or the networking stack below the client. Further, WPAD will need to occur in response to expiration of existing configuration data. The following sections describe the details of these scenarios.

The web proxy auto-discovery process MUST occur at least as frequently as one of the following two options. A web client can use either option depending on which makes sense in their environment. Clients MUST use at least one of the following options. They MAY also choose to implement both options.

- o Upon startup of the web client
- o Whenever there indication from the networking stack that the IP address of the client host either has, or could have, changed

In addition, the client MUST attempt a discovery cycle upon expiration of a previously downloaded CFILE in accordance with HTTP/1.1[15].

5.2.1 Upon Startup of the Web Client

For many types of web client (like web browsers) there can be many instances of the client operating for a given user at one time. This is often to allow display of multiple web pages in different windows, for example. There is no need to re-perform WPAD every time a new instance of the web client is opened. WPAD MUST be performed when the number of web client instances transitions from 0 to 1. It SHOULD NOT be performed as additional instances are created.

5.2.2 Network Stack Events

Another option for clients is to tie the execution of WPAD to changes in the networking environment. If the client can learn about the change of the local host's IP address, or the possible change of the IP address, it MUST re-perform the WPAD protocol. Many operating systems provide indications of "network up" events, for example. Those types of events and system-boot events might be the triggers for WPAD in many environments.

5.2.3 Expiration of the CFILE

The HTTP retrieval of the CURL may return HTTP headers specifying a valid lifetime for the CFILE returned. The client MUST obey these timeouts and rerun the WPAD process when it expires. A client MAY rerun the WPAD process if it detects a failure of the currently configured proxy (which is not otherwise recoverable via the servers provide support for making configuration decisions based on subnets, which are directly related to network topology.

Full client support for DHCP is not as ubiquitous as for DNS. That is, not all clients are equipped to take advantage of DHCP for their essential network configuration (assignment of IP address, network mask, etc). APIs for DHCP are not as widely available. Luckily, using DHCP for WPAD does not require either of these facilities. It is relatively easy for web client developers to speak just the minimal DHCP protocol to perform resource discovery. It entails building a simple UDP packet, sending it to the subnet broadcast address, and parsing the reply UDP packet(s) which are received to extract the WPAD option field. A reference implementation of this code in C is available [12].

The WPAD client attempts a series of resource discovery requests, using the discovery mechanisms mentioned above, in a specific order. Clients only attempt mechanisms that they support (obviously). Each time the discovery attempt succeeds, the client uses the information obtained to construct a CURL. If a CFILE is successfully retrieved at that CURL, the process completes. If not, the client resumes where it left off in the predefined series of resource discovery requests. If no untried mechanisms remain and a CFILE has not been successfully retrieved, the WPAD protocol fails and the client is configured to use no proxy.

First the client tries DHCP, followed by SLP. If no CFILE has been retrieved the client moves on to the DNS based mechanisms. The client will cycle through the DNS SRV, "Well Known Aliases" and DNS TXT record methods multiple times. Each time through the QNAME being used in the DNS query is made less and less specific. In this manner the client can locate the most specific configuration information possible, but can fall back on less specific information. Every DNS lookup has the QNAME prefixed with "wpad" to indicate the resource type being requested.

As an example, consider a client with hostname johns-desktop.development.foo.com. Assume the web client software supports all of the mechanisms listed above. This is the sequence of discovery attempts the client would perform until one succeeded in locating a valid CFILE:

o DHCP
o SLP
o DNS A lookup on QNAME=wpad.development.foo.com.
o DNS SRV lookup on QNAME=wpad.development.foo.com.
o DNS TXT lookup on QNAME=wpad.development.foo.com.
o DNS A lookup on QNAME=wpad.foo.com.
o DNS SRV lookup on QNAME=wpad.foo.com.
o DNS TXT lookup on QNAME=wpad.foo.com.

```
         slp_list = slp_list.next;
   }

   /* all the DNS mechanisms */
   TGTDOM = gethostbyname(me);
   TGTDOM = strip_leading_component(TGTDOM);

   while (is_not_canonical(TGTDOM)) {

      /* SHOULD try DNS SRV records */
      dns_list = dns_query(/*QNAME=wpad.TGTDOM.,
                             QTYPE=SRV (Section 5.4.4)*/);
      while (dns_list != null) { /* each TXT record */
         if isvalid(read_CFILE(dns_list, curl_data))
            return SUCCESS;   /* valid CFILE */
         else
            dns_list = dns_list.next;
      }

      /* SHOULD try DNS TXT records */
      dns_list = dns_query(/*QNAME=wpad.TGTDOM.,
                             QTYPE=TXT (Section 5.4.5)*/);
      while (dns_list != null) { /* each TXT record */
         if isvalid(read_CFILE(dns_list, curl_data))
            return SUCCESS;   /* valid CFILE */
         else
            dns_list = dns_list.next;
      }

      /* MUST try DNS A records */
      dns_list = dns_query(/*QNAME=wpad.TGTDOM.,
                         QTYPE=A  (Section 5.4.3)*/);

      while (dns_list != null) { /*check each A record */
         if isvalid(read_CFILE(dns_list, curl_data))
            return SUCCESS;   /* valid CFILE */
         else
            dns_list = dns_list.next;
      }

      /* Still no match, remove leading component and iterate */
      TGTDOM = strip_leading_component(TGTDOM);

   } /* no A, TXT or SRV records for wpad.* */

   return FAILED;  /* could not locate valid CFILE */
}
```

inherent mechanisms provided by the currently active Configuration File).

Whenever the client decides to invalidate the current CURL or CFILE, it MUST rerun the entire WPAD protocol to ensure it discovers the currently correct CURL. Specifically, if the valid lifetime of the CFILE ends (as specified by the HTTP headers provided when it was retrieved), the complete WPAD protocol MUST be rerun. The client MUST NOT simply re-use the existing CURL to obtain a fresh copy of the CFILE.

A number of network round trips, broadcast and/or multicast communications may be required during the WPAD protocol. The WPAD protocol SHOULD NOT be invoked at a more frequent rate than specified above (such as per-URL retrieval).

5.3 WPAD Protocol Specification

The following pseudo-code defines the WPAD protocol. If a particular discovery mechanism is not supported, treat it as a failed discovery attempt in the pseudo-code.

Two subroutines need explanation. The subroutine strip_leading_component(dns_string) strips off the leading characters, up to and including the first dot ('.') in the string which is passed as a parameter, and is expected to contain DNS name. The Boolean subroutine is_not_canonical(dns_string) returns FALSE if dns_string is one of the canonical domain suffixes defined in RFC 1591[13] (for example, "com").

The slp_list and dns_list elements below are assumed to be linked lists containing a data field and a pointer to the next element. The data field contains the elements used to override the default values in creating a CURL, as detailed in Section 5.5.

```
load_CFILE() {
/* MUST use DHCP */
curl = dhcp_query(/*WPAD option (Section 5.4.1) */);
if (curl != null) {   /* DHCP succeeded */
   if isvalid (read_CFILE(curl))
      return SUCCESS;   /* valid CFILE */
}

/* Should use SLP */
slp_list = slp_query(/*(WPAD attributes  (Section 5.4.2)*/);
while (slp_list != null) {  /* test each curl   */
   if isvalid(read_CFILE(slp_list.curl_data))
      return SUCCESS;   /* valid CFILE */
   else
```

5.4.2 Service Location Protocol /SLP

The Service Location Protocol[14] is a Proposed Standard for discovering services in the Internet. SLP has several reference implementations available; for details, check [16].

A service type for use with WPAD has been defined and is available as an Internet Draft.

Client implementations SHOULD implement SLP. SLP Service Replies will provide one or more complete CURLs. Each candidate CURL so created should be pursued as specified in Section 5.5 and beyond.

5.4.3 DNS A/CNAME "Well Known Aliases"

Client implementations MUST support this mechanism. This should be straightforward since only basic DNS lookup of A records is required. See RFC 2219[6] for a description of using "well known" DNS aliases for resource discovery. We propose the "well known alias" of "wpad" for web proxy auto-discovery.

The client performs the following DNS lookup:
QNAME=wpad.TGTDOM., QCLASS=IN, QTYPE=A

Each A RR, which is returned, contains an IP address which is used to replace the <HOST> default in the CURL.

Each candidate CURL so created should be pursued as specified in Section 5.5 and beyond.

5.4.4 DNS SRV Records

Client implementations SHOULD support the DNS SRV mechanism. Details of the protocol can be found in RFC 2052[2]. If the implementation language/environment provides the ability to perform DNS lookups on QTYPEs other than A, client implementers are strongly encouraged to provide this support. It is acknowledged that not all resolver APIs provide this functionality.

The client issues the following DNS lookup:
QNAME=wpad.tcp.TGTDOM., QCLASS=IN, QTYPE=SRV

If it receives SRV RRs in response, the client should use each valid RR in the order specified in RFC 2052[2]. Each valid record will specify both a <HOST> and <PORT> to override the CURL defaults.

Each candidate CURL so created should be pursued as specified in Section 5.5 and beyond.

5.4 Discovery Mechanisms

Each of the resource discovery methods will be marked as to whether the client MUST, SHOULD, MAY, or MUST NOT implement them to be compliant. Client implementers are encouraged to implement as many mechanisms as possible, to promote maximum interoperability.

SUMMARY OF DISCOVERY MECHANISMS

Discovery Mechanism	Status	Document Section
DHCP	MUST	5.4.1
SLP	SHOULD	5.4.2
"Well Known Alias"	MUST	5.4.3
DNS SRV Records	SHOULD	5.4.4
DNS TXT "service: URLs"	SHOULD	5.4.5

5.4.1 DHCP

Client implementations MUST support DHCP. DHCP has widespread support in numerous vendor hardware and software implementations, and is widely deployed. It is also perfectly suited to this task, and is used to discover other network resources (such a time servers, printers, etc). The DHCP protocol is detailed in RFC 2131[3]. We propose a new DHCP option with code 252 for use in web proxy auto-discovery. See RFC 2132[4] for a list of existing DHCP options. See "Conditional Compliance" (Section 9) for more information on DHCP requirements.

The client should obtain the value of the DHCP option code 252 as returned by the DHCP server. If the client has already conducted DHCP protocol during its initialization, the DHCP server may already have supplied that value. If the value is not available through a client OS API, the client SHOULD use a DHCPINFORM message to query the DHCP server to obtain the value.

The DHCP option code for WPAD is 252 by agreement of the DHC working group chair. This option is of type STRING. This string contains a URL which points to an appropriate config file. The STRING is of arbitrary size.

An example STRING value would be:
"http://server.domain/proxyconfig.pac"

(perhaps in the form of an IP address). Some mechanisms will also provide a <PORT> and/or a <PATH>. The client should override the default CURL fields with all of those supplied by the discovery mechanism.

5.6 Retrieving the CFILE at the CURL

The client then requests the CURL via HTTP. When making the request it MUST transmit HTTP "Accept" headers indicating what CFILE formats it is capable of accepting. For example, Netscape Navigator browsers with versions 2.0 and beyond might include the following line in the HTTP Request:

```
Accept: application/x-ns-proxy-autoconfig
```

The client MUST follow HTTP redirect directives (response codes 3xx) returned by the server. The client SHOULD send a valid "User-Agent" header.

5.7 Resuming Discovery

If the HTTP request fails for any reason (fails to connect, server error response, etc.) the client MUST resume the search for a successful CURL where it left off. It should continue attempting other sub-steps in a specific discovery mechanism, and then move on to the next mechanism or TGTDOM iteration, etc.

6. Client Implementation Considerations

The large number of discovery mechanisms specified in this document may raise concerns about network traffic and performance. The DHCP portion of the process will result in a single broadcast by the client, and perhaps a few replies by listening DHCP servers.

The remaining mechanisms are all DNS based. All DNS queries should have the QNAME terminated with a trailing '.' to indicate a FQDN and expedite the lookup. As such each TGTDOM iteration will cause 3 DNS lookups, each a unicast UDP packet and a reply. Most clients will have fewer than 2 TGTDOM iterations, limiting the total number of DNS request/replies to 6.

In total, 7 UDP request/reply packets on client startup is quite a low overhead. The first web page downloaded by the client will likely dwarf that packet count. Each of the DNS lookups should stand a high chance of hitting the cache in the client's DNS server, since other clients will have likely looked them up recently, providing a low total elapsed time.

This is of course the worst case, where no CURLS are obtained, and

5.4.5 DNS TXT service: Entries

Client implementations SHOULD support this mechanism. If the implementation language/environment provides the ability to perform DNS lookups on QTYPEs other than A, the vendor is strongly encouraged to provide this support. It is acknowledged that not all resolver APIs provide this functionality.

The client should attempt to retrieve TXT RRs from the DNS to obtain "service: URLs" contained therein. The "service: URL" will be of the following format, specifying a complete candidate CURL for each record located:

```
service: wpad:http://<HOST>:<PORT><PATH>
```

The client should first issue the following DNS query:
QNAME=wpad.TGTDOM., QCLASS=IN, QTYPE=TXT

It should process each TXT RR it receives (if any) using each service:URL found (if any) to generate a candidate CURL. These CURLs should be pursued as described in Section 5.5 and beyond. Readers familiar with [1] should note that WPAD clients MUST NOT perform the QNAME=TGTDOM., QCLASS=IN, QTYPE=TXT lookup which would be suggested by that document.

5.4.6 Fallback

Clients MUST NOT implement the "Fallback" mechanism described in [1]. It is unlikely that a client will find a web server prepared to handle the CURL request at a random suffix of its FQDN. This will only increase the number of DNS probes and introduce an excess of spurious "GET" requests on those hapless web servers.

Instead, the "Well Known Aliases" method of Section 5.4.4 provides equivalent functionality.

5.4.7 Timeouts

Implementers are encouraged to limit the time elapsed in each discovery phase. When possible, limiting each phase to 10 seconds is considered reasonable. Implementers may choose a different value which is more appropriate to their network properties. For example, a device implementation, which operated over a wireless network, may use a much larger timeout to account for low bandwidth or high latency.

5.5 Composing a Candidate CURL

Any successful discovery mechanism response will provide a <HOST>

o The web server handling the CURL request can make decisions based on the "User-Agent", "Accept", client IP address/subnet/hostname, and the topological distribution of nearby proxies, etc. This can occur inside a CGI executable created to handle the CURL. As mentioned above it could be a proxy server itself handing the CURL request and making those decisions.

o The CFILE may be expressive enough to select from a set of alternatives at "runtime" on the client. CARP[10] is based on this premise for an array of caches. It is not inconceivable that the CFILE could compute some network distance or fitness metrics to a set of candidate proxies and then select the "closest" or "most responsive" device.

Note that it is valid to configure a DHCP daemon to respond only to INFORM option queries in static IP environments

Not all of the above mechanisms can be supported in all currently deployed vendor hardware and software. The hope is that enough flexibility is provided in this framework that administrators can select which mechanisms will work in their environments.

9. Conditional Compliance

In light of the fact that many of the discovery technologies described in this document are not well deployed or not available on all platforms, this specification permits conditional compliance. Conditional compliance is designated by three class identifications.

Additionally, due to the possible security implications of a DHCP broadcast request, it is onerous to REQUIRE an implementer to put himself or his implementation at undue risk. It is quite common to have rogue DHCP servers on a network which may fool a DHCP broadcast implementation into using a malicious configuration file. On platforms which do not support DHCP natively and cannot get the WPAD option along with its IP address, and which cannot support the DHCP INFORM unicast request, presumably to a known and trusted DHCP server, the likelihood of an undetected spoofing attack is increased. Having an individual program, such as a browser, trying to detect a DHCP server on a network is unreasonable, in the authors' opinion. On platforms which use DHCP for their system IP address and have previously trusted a DHCP server, a unicast DHCP INFORM to that same trusted server does not introduce any additional trust to that server.

9.1 Class 0 - Minimally compliant

A WPAD implementation which implements only the following discovery mechanisms and interval schemes is considered class 0 compliant:

DNS A record queries

Browser or System session refresh intervals

Class 0 compliance is only applicable to systems or implementations which do not natively support DHCP and/or cannot securely determine a trusted local DHCP server.

9.2 Class 1 - Compliant

A WPAD implementation which implements only the following discovery mechanisms and interval schemes is considered class 1 compliant:

DNS A record queries
DHCP INFORM Queries
Network stack change refresh intervals
CFILE expiration refresh intervals

9.3 Class 2 - Maximally compliant

A WPAD implementation which implements only the following discovery mechanisms and interval schemes is considered class 1 compliant:

DNS A record queries
DHCP INFORM Queries
DNS TXT service: queries
DNS SRV RR queries
SVRLOC Queries
Network stack change refresh intervals
CFILE expiration refresh intervals

To be considered compliant with a given class, an implementation MUST support the features listed above corresponding to that class.

10. Security Considerations

This document does not address security of the protocols involved. The WPAD protocol is vulnerable to existing identified weaknesses in DHCP and DNS. The groups driving those standards, as well as the SLP protocol standards, are addressing security.

When using DHCP discovery, clients are encouraged to use unicast DHCP INFORM queries instead of broadcast queries which are more easily spoofed in insecure networks.

Minimally, it can be said that the WPAD protocol does not create new security weaknesses.

11. Acknowledgements

The authors' work on this specification would be incomplete without the assistance of many people. Specifically, the authors would like the express their gratitude to the following people:

Chuck Neerdaels, Inktomi, for providing assistance in the design of the WPAD protocol as well as for providing reference implementations.

Arthur Bierer, Darren Mitchell, Sean Edmison, Mario Rodriguez, Danpo Zhang, and Yaron Goland, Microsoft, for providing implementation insights as well as testing and deployment.

Ari Luotonen, Netscape, for his role in designing the first web proxy.

In addition, the authors are grateful for the feedback provided by the following people:

o Jeremy Worley (RealNetworks)
o Eric Twitchell (United Parcel Service)

References

[1] Moats, R., Hamilton, M. and P. Leach, "Finding Stuff (How to discover services) (Internet Draft)", October 1997.

[2] Gulbrandsen, A. and P. Vixie, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2052, October 1996, <URL:http://www.ietf.org/rfc/rfc2052.txt>.

[3] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997, <URL:http://www.ietf.org/rfc/rfc2131.txt>.

[4] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, March 1997, <URL:http://www.ietf.org/rfc/rfc2131.txt>.

[5] Veizades, J., Guttman, E., Perkins, C. and M. Day, "Service Location Protocol (Internet Draft)", October 1997.

[6] Hamilton, M. and R. Wright, "Use of DNS Aliases for Network Services", RFC 2219, October 1997, <URL:http://www.ietf.org/rfc/rfc2219.txt>.

[7] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997, <URL:http://www.ietf.org/rfc/rfc2119.txt>.

[8] Luotonen, A., "Navigator Proxy Auto-Config File Format", March 1996, <URL:http://home.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html>.

[9] Mockapetris, P., "Domain Names - Concepts and Facilities", RFC

1034, November 1987,
<URL:http://www.ietf.org/rfc/rfc1034.txt>.

[10] Valloppillil, V. and K.W. Ross, "Cache Array Routing Protocol", draft-vinod-carp-v1-03.txt (work in progress), February 1998,
<URL:http://www.wrec.org/Drafts/draft-vinod-carp-v1-03.txt>.

[11] Perkins, C., Guttman, E. and J. Kempf, "Service Templates and service: Schemes (Internet Draft)", December 1997.

[12] "A Sample DHCP Implementation for WPAD", February 1998,
<URL:http://www.inktomi.com/TBD.html>.

[13] Postel, J., "Domain Name System Structure and Delegation", RFC 1591, March 1994,
<URL:http://www.ietf.org/rfc/rfc1591.txt>.

[14] Guttman, E., Perkins, C., Viezades, J. and M. Day, "Service Location Protocol, Version 2", RFC 2608, June 1999,
<URL:http://www.ietf.org/rfc/rfc2608.txt>.

[15] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999,
<URL:http://www.ietf.org/rfc/rfc2616.txt>.

[16] <URL:http://www.srvloc.org/>

Authors' Addresses

Ian Cooper
Equinix, Inc.

EMail: icooper@equinix.com

Paul Gauthier
Inktomi Corporation

EMail: gauthier@inktomi.com

Josh Cohen
(Microsoft Corporation)

Martin Dunsmuir
(RealNetworks, Inc.)

Charles Perkins
Sun Microsystems, Inc.

EMail: charles.perkins@Sun.COM

Full Copyright Statement

Acknowledgement

Funding for the RFC editor function is currently provided by the Internet Society.

This Page Blank (uspto)

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- □ BLACK BORDERS
- □ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☑ FADED TEXT OR DRAWING
- ☒ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- □ SKEWED/SLANTED IMAGES
- □ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- □ GRAY SCALE DOCUMENTS
- □ LINES OR MARKS ON ORIGINAL DOCUMENT
- □ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- □ OTHER: ______________________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

This Page Blank (uspto)

(H) Art submitted by Applicant - Entered

The following items (1) – (2) listed below are hereby entered as evidence submitted to the Examiner as noted on an IDS transmission received by the USPTO on February 12, 2001.

(1) Copy of U.S. Patent Number 5,790,977 ("Ezekiel"). This evidence was entered into the record by the Applicant on 02/07/2001.

(2) Copy of U.S. Patent Number 5,991,795 ("Howard et al."). This evidence was entered into the record by the Applicant on 02/07/2001.

Copies of all References follows.

//

US005790977A

United States Patent [19]

Ezekiel

[11] **Patent Number: 5,790,977**

[45] **Date of Patent: Aug. 4, 1998**

[54] **DATA ACQUISITION FROM A REMOTE INSTRUMENT VIA THE INTERNET**

[75] Inventor: **David Ezekiel**, Santa Rosa, Calif.

[73] Assignee: **Hewlett-Packard Company**, Palo Alto, Calif.

[21] Appl. No.: **795,443**

[22] Filed: **Feb. 6, 1997**

[51] **Int. Cl.**^{6} **G06F 17/00**

[52] **U.S. Cl.** **702/122**; 395/200.48; 395/200.59; 380/49; 380/25

[58] **Field of Search** 364/579; 395/200.47, 395/200.49, 200.33, 200.57, 187.01, 188.01, 500, 200.48, 200.59; 380/4, 25, 49

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,062,059	10/1991	Youngblood et al.	395/200.47
5,349,678	9/1994	Morris et al.	395/200.49
5,375,207	12/1994	Blakely et al.	395/200.33
5,544,320	8/1996	Konrad	395/200.33
5,553,239	9/1996	Heath et al.	395/187.01
5,657,390	8/1997	Elgamal et al.	380/49
5,708,780	1/1998	Levergood et al.	1/1

OTHER PUBLICATIONS

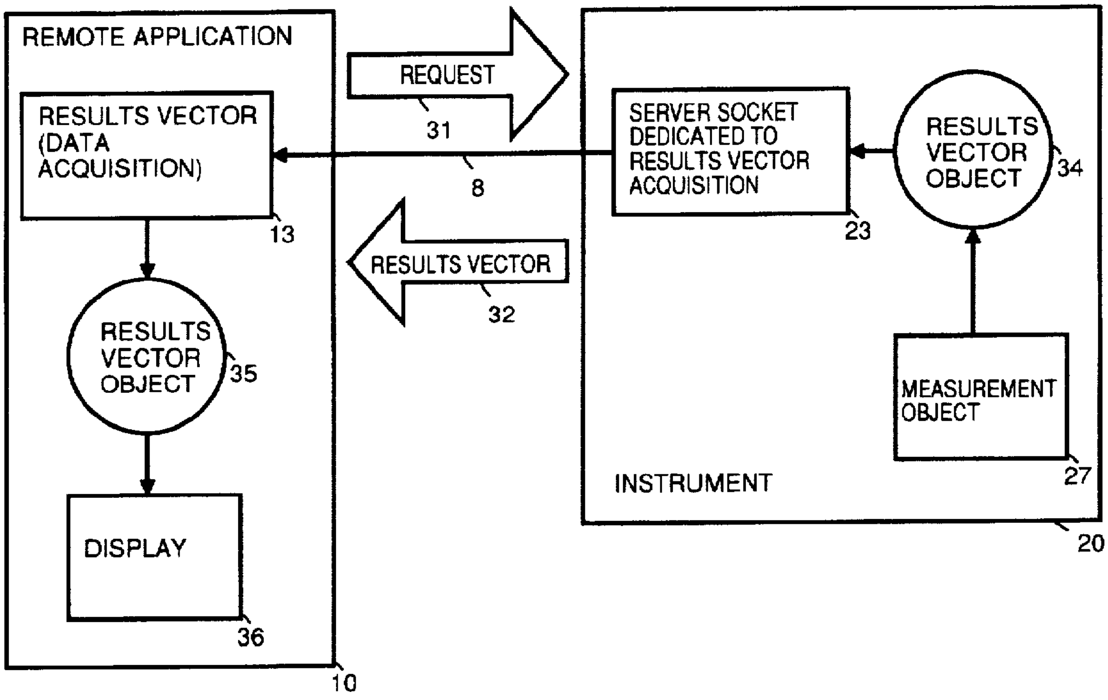
Bob Cutler, "LAN in an Instrument: Two Years of Practical Use": A four page unpublished paper which describes use of an LAN option for the HP89400 series of vector signal analysers.

Primary Examiner—John E. Barlow, Jr.
Assistant Examiner—Bryan Bui

[57] **ABSTRACT**

Remote access is provided from a remote host system to an instrument. Control and data acquisition software is stored within the instrument. In response to the remote host system, the control and data acquisition software is forwarded to the remote host system. The control and data acquisition software is run on the remote host system. In response to control commands from the control and data acquisition software running on the remote host system, data acquisition of the instrument is controlled. In response to a request from the control and data acquisition software running on the remote host system, acquired data is forwarded from the instrument to the remote host system.

21 Claims, 5 Drawing Sheets



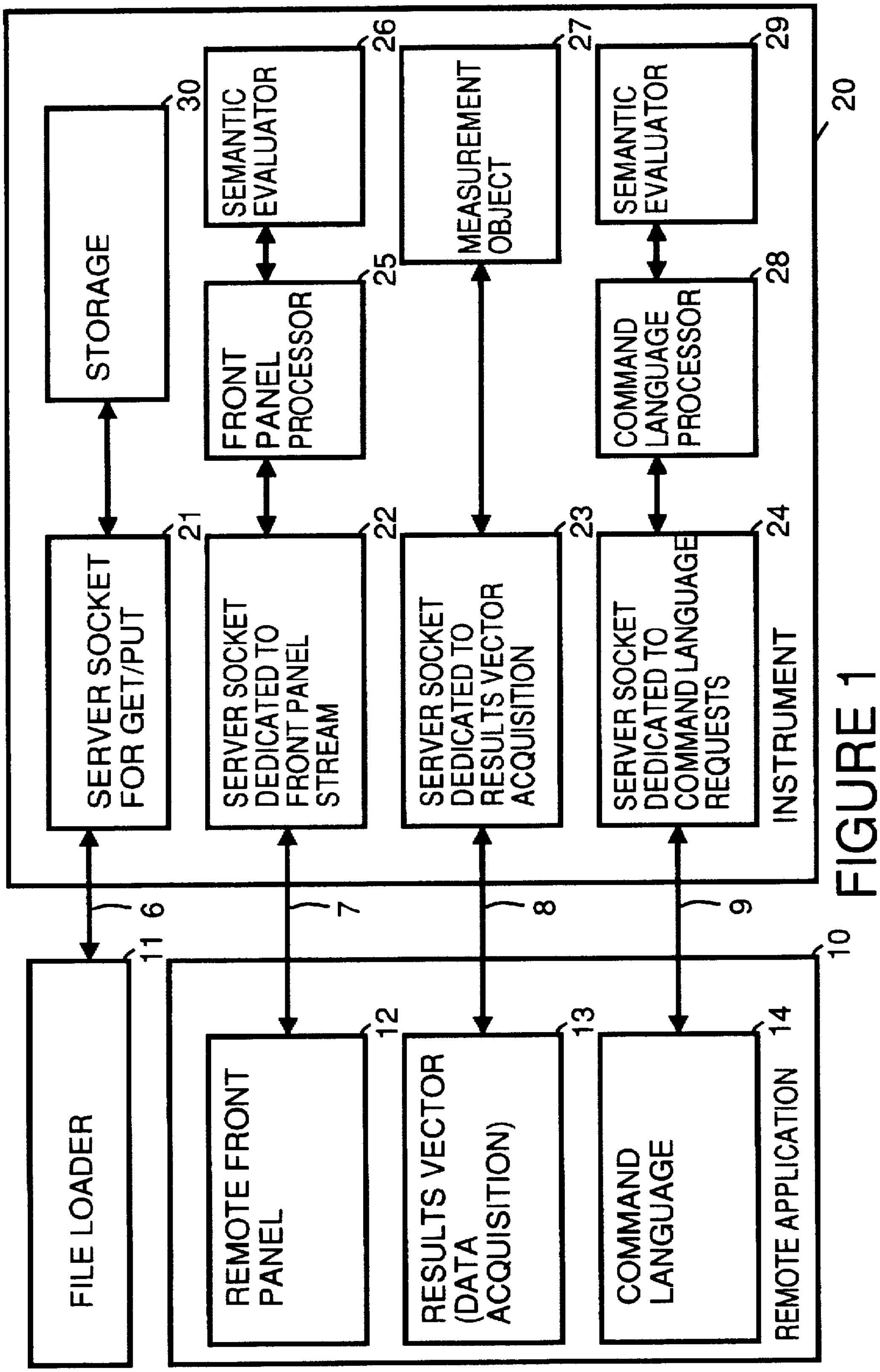
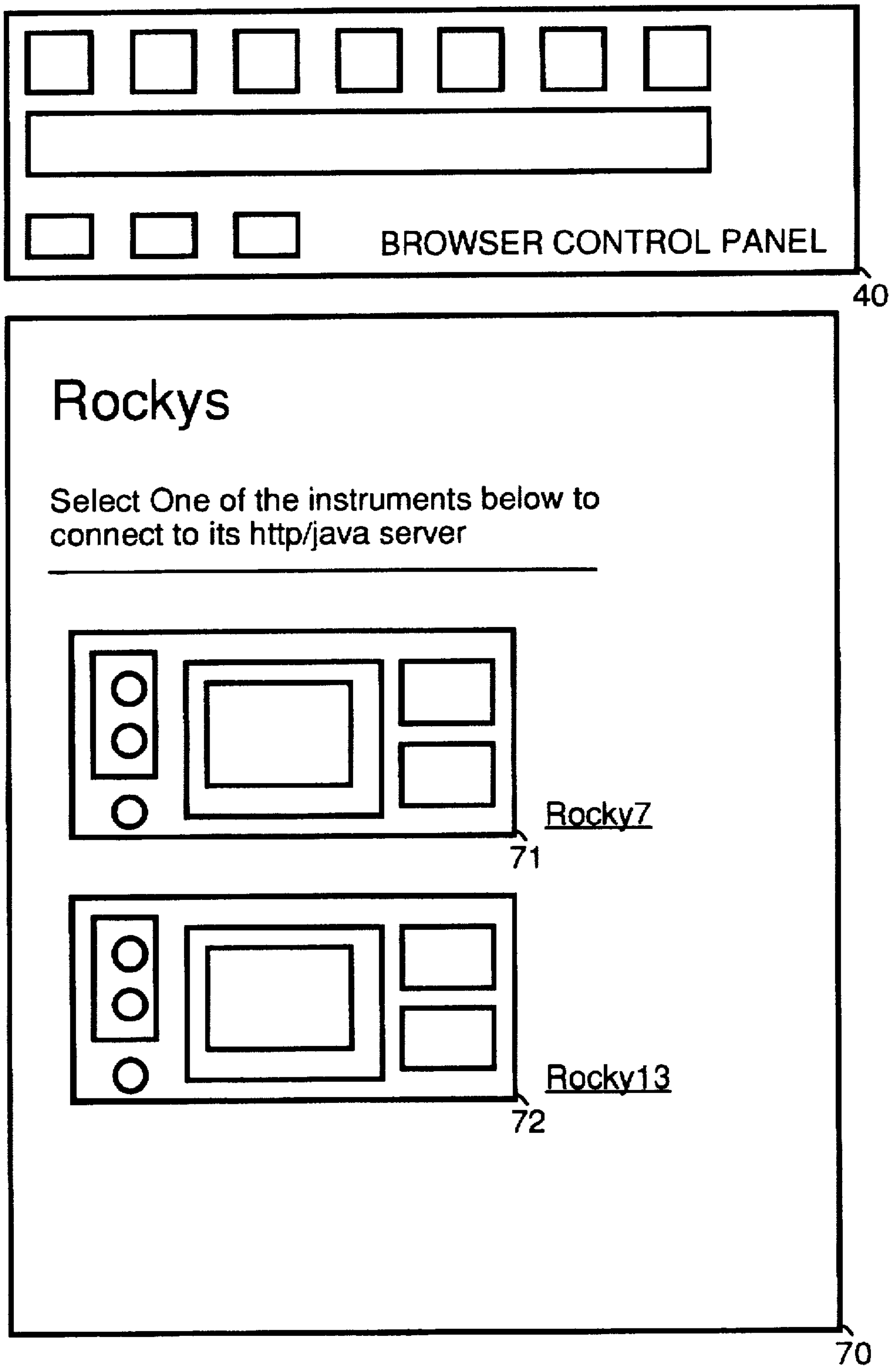

STORAGE
30
SEMANTIC EVALUATOR
26
MEASUREMENT OBJECT
27
SEMANTIC EVALUATOR
29
20
FRONT PANEL PROCESSOR
25
COMMAND LANGUAGE PROCESSOR
28
SERVER SOCKET FOR GET/PUT
21
SERVER SOCKET DEDICATED TO FRONT PANEL STREAM
22
SERVER SOCKET DEDICATED TO RESULTS VECTOR ACQUISITION
23
SERVER SOCKET DEDICATED TO COMMAND LANGUAGE REQUESTS
24
INSTRUMENT
6
7
8
9
10
11
FILE LOADER
REMOTE FRONT PANEL
12
RESULTS VECTOR (DATA ACQUISITION)
13
COMMAND LANGUAGE
14
REMOTE APPLICATION

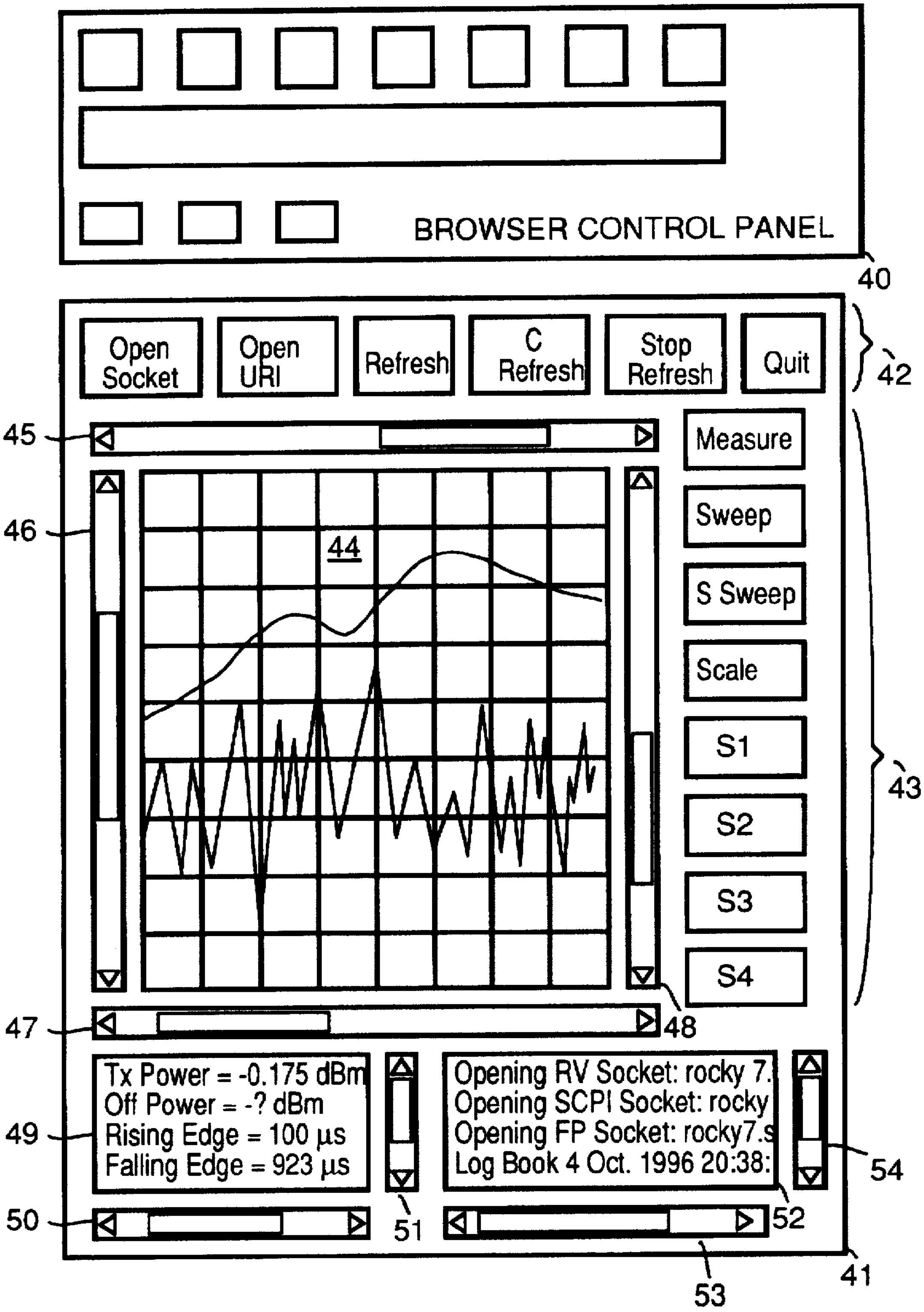
FIGURE 1



Rockys

Select One of the instruments below to connect to its http/java server

FIGURE 2



Tx Power = -0.175 dBm
Off Power = -? dBm
Rising Edge = 100 µs
Falling Edge = 923 µs

Opening RV Socket: rocky 7.
Opening SCPI Socket: rocky
Opening FP Socket: rocky7.s
Log Book 4 Oct. 1996 20:38:

FIGURE 3

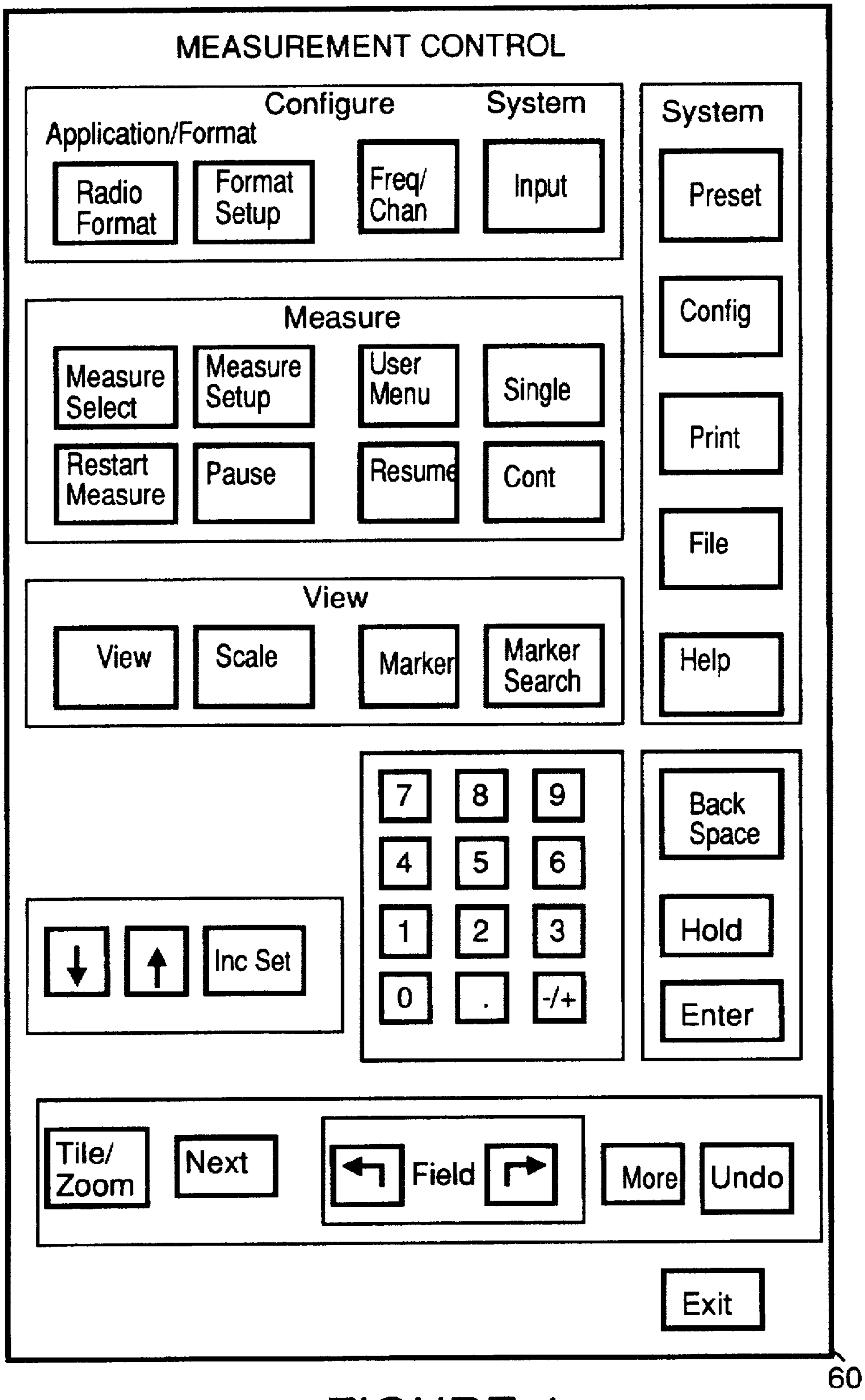

MEASUREMENT CONTROL
Configure
System
Application/Format
Radio Format
Format Setup
Freq/ Chan
Input
System
Preset
Config
Print
File
Help
Measure
Measure Select
Measure Setup
User Menu
Single
Restart Measure
Pause
Resume
Cont
View
View
Scale
Marker
Marker Search
7
8
9
4
5
6
1
2
3
0
.
-/+
Back Space
Hold
Enter
↓
↑
Inc Set
Tile/ Zoom
Next
Field
More
Undo
Exit
60

FIGURE 4

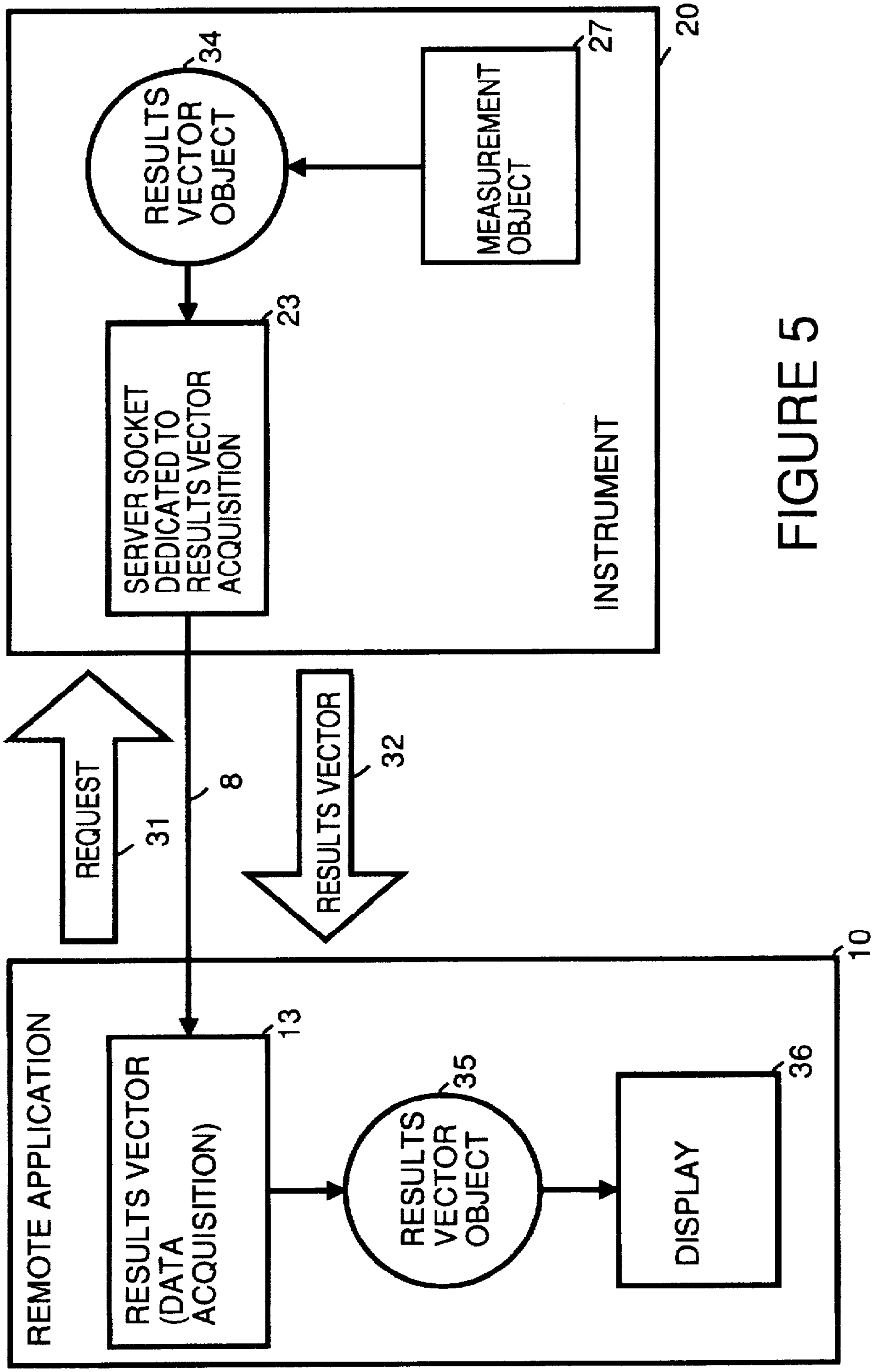

RESULTS VECTOR OBJECT
34
MEASUREMENT OBJECT
27
20
SERVER SOCKET DEDICATED TO RESULTS VECTOR ACQUISITION
23
INSTRUMENT
REQUEST
31
8
RESULTS VECTOR
32
10
REMOTE APPLICATION
RESULTS VECTOR (DATA ACQUISITION)
13
RESULTS VECTOR OBJECT
35
DISPLAY
36

FIGURE 5

DATA ACQUISITION FROM A REMOTE INSTRUMENT VIA THE INTERNET

BACKGROUND

The present invention concerns communications with analytical instruments and pertains particularly to the data acquisition from a remote instrument, for example using a network browser on the World Wide Web.

In recent years there has been progress in both telecommunications technology and computing technology. Through the proliferation of on-line services and internet services, a wealth of information is available to a user having access to a personal computer and a modem. In order to allow users of the internet to take advantage of the explosion of information available on the internet, as well as a catalyst for information being made available, a number of hardware and software products have been developed.

For example, web browsers available from various companies allow users to "surf" within the World Wide Web. The Netscape Navigator web browser, for example, is available from Netscape Communications Corporation. Use of web browsers over the World Wide Web facilitates the communication of graphics, audio and video data, as well as text.

In order to reduce the software incompatibility of information communicated over the internet, various languages have been proposed for development of applications which utilize the world wide web. For example, the JAVA programming language, available from Sun Microsystems, is increasingly used to develop applications which communicate over the World Wide Web. Theoretically, use of the JAVA programming language allows for smooth and versatile communication between computing systems even when the computing systems are operating on different hardware platforms and are using different operating systems.

As the ability and versatility of communication over the internet increases, there is much consideration and developmental effort being invested to determine new and innovative ways to take advantage of this rapidly developing technology.

SUMMARY OF THE INVENTION

In accordance with the preferred embodiment of the present invention, remote access is provided from a remote host system to an instrument. Control and data acquisition software is stored within the instrument. In response to the remote host system, the control and data acquisition software is forwarded to the remote host system. The control and data acquisition software is run on the remote host system. In response to control commands from the control and data acquisition software running on the remote host system, data acquisition of the instrument is controlled. In response to a request from the control and data acquisition software running on the remote host system, acquired data is forwarded from the instrument to the remote host system.

For example, in the preferred embodiment, the control and data acquisition software runs as an application within an internet browser. In this case, the control and data acquisition software can be forwarded to the remote host system in response to a HyperText Transfer Protocol (HTTP) server GET command.

In the preferred embodiment of the present invention, when the control and data acquisition software is run on the remote host system, a user of the remote host system is provided with a graphical user interface which the user can utilize to interact with the instrument. The data received from the instrument by the remote host system is displayed on the host system, for example as traces on a graticule. The data may be additionally processed at the remote host system before being displayed.

The present invention provides for remote data acquisition from an instrument to be performed over the internet. Using the present invention, client software can run on any internet browser on almost all platforms. Transparent to the user, software used to run the instrument can be updated. The present invention additionally allows for the partitioning of the display and sensor portions of an instruments which allows for a significant reduction in the cost of instrument hardware and higher performance of the instrument as client platforms increase in capability in the installed base. The present invention also allows for instruments to access other instruments, directly controlling and acquire data from each other over a network. Further, for systems applications, a single client can connect to many instruments.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows a block diagram of an instrument in communication with a remote application in accordance with a preferred embodiment of the present invention.

FIG. **2** shows links to instruments in a web page as displayed by the remote application shown in FIG. **1** in accordance with a preferred embodiment of the present invention.

FIG. **3** shows a simplified display of data acquired by the remote application from the instrument shown in FIG. **1** in accordance with a preferred embodiment of the present invention.

FIG. **4** shows a simplified display of a virtual front panel acquired by the remote application from the instrument shown in FIG. **1** in accordance with a preferred embodiment of the present invention.

FIG. **5** is a block diagram which illustrates data acquisition from the instrument to the remote application shown in FIG. **1** in accordance with a preferred embodiment of the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. **1** shows a block diagram of an instrument **20** in communication with a remote application **10** in accordance with a preferred embodiment of the present invention. Remote application **10**, is for example, a Java applet application which runs within a Web browser application. Instrument **20** is, for example, a vector modulation analyzer instrument. Utilizing client/server terminology, the web browser application is the client and instrument **20** is the server.

In the preferred embodiment, remote application **10** includes a remote front panel module **12**, a results vector module **13** and a command language module **14**.

Remote front panel module **12** provides for remote control of instrument **20** via display of and user interaction with a graphical user interface which represents front panel controls of instrument **20**. Results vector module **13** provides for data acquisition from instrument **20** for real-time dynamic display by remote application **10** to a user. Command language module **14** provides a remote command language request/response scheme for status, control and so on. For example, command language module **14** utilizes the Standard Commands for Programmable Instruments (SCPI) command language (IEEE 488.1, IEEE 488.2).

A file loader **11** is used to load files to and from instrument **20**. For example, files are loaded over a port **6** using the HyperText Transfer Protocol (HTTP) GET/PUT commands. Alternatively, this may be done, for example, by extending the functionality of remote application **10** to perform HTTP GET/PUT commands. In the preferred embodiment of the present invention, port **6** corresponds to port **8080** on a Netscape Navigator web browser, available from Netscape Communications Corporation.

Instrument **20** is connected over the internet to a web browser using four sockets. For example, each socket is a two-way application communication channel initiated from the client to the server using industry standard socket protocol. A server socket **21** is used as a general HTTP server for responding to GET/PUT commands. In response to GET commands, server socket loads files from storage **30** to file loader **11**. In response to PUT commands, server socket puts files from file loader **11** into the internal file system (within storage **30**) of instrument **20**.

A server socket **22** is dedicated to receiving requests for pushing keys on a front panel of instrument **20**. When a user interacts with a graphic user interface displayed by remote application **10** by, for example, selecting a specific key displayed by the graphic user interface, remote front panel **12** forwards to server socket **22** over a port **7** a predetermined numeric ("keycode") which is defined to correspond to a particular control function of the instrument. Upon receiving such a keycode, server socket **22** forwards the keycode to front panel processor **25**, which, with the help of a semantic evaluator **26**, performs the specified control function. For example, the keycodes are generated and transferred in accordance with a JAVA front panel (FP) keycode protocol.

A server socket **23** is dedicated to retrieving data and controlling transforms from results vectors produced by a measurement object **27** within instrument **20**. For example, the results vectors are transferred over a port **8** in response to a JAVA results vector (RV) request. Transform control is added to this protocol to allow for the specification of transform settings in the results vector at pre-measurement and post-measurement data acquisition periods within the numeric data format of results vectors.

A server socket **24** is dedicated to receiving and returning command language requests over a port **9** in accordance with predetermined command strings used for control, status and data acquisition. Upon receiving a command language request, server socket **24** forwards the command language to command panel processor **28**, which with the help of a semantic evaluator **29** performs the specified command function. For example, the command language requests are SCPI language requests which are compliant with SCPI IEEE 488.2. SCPI IEEE 488.2 is an instrument industry standard protocol extended for the Vector Modulation Analyzer. When instrument **20** is an instrument other than a vector modulation analyzer instrument, another SCPI protocol or other command language can be used. Results passed from instrument **20** to remote application **10** are, for example, in standard SCPI protocol.

For example, server sockets **22**, **23** and **24** are C and C++ dedicated socket services. Each of server sockets **22**, **23** and **24** continue to provide socket communication during an entire client browser session.

FIG. **2** shows a browser control panel **40**. In addition, FIG. **2** shows a HyperText MarkUpLanguage (HTML) web page **70** as displayed by remote application **10**, shown in FIG. **1**. A Uniform Resource Locator (URL) **71** links remote application **10** to instrument **20**. The application class of instrument **20** is linked to web page 70 with an HTTP address made up of the host name (i.e., "Rocky7") of instrument **20** or internet Transmission Control Protocol (TCP)/Internet Protocol (IP) address (e.g., 15.8.162.231), and the remote application port (e.g. port **8080**) used for retrieving remote application **10**, shown in FIG. **1**. For example, the following line in the HTML application language connects a web page to instrument **20**:

```
<A HREF="http://rocky7:8080">
```

A Uniform Resource Locator (URL) **72** links remote application **10** to a second instrument. The application class of the second instrument is linked to web page 70 with an HTTP address made up of the host name (i.e., "Rocky13") of instrument **20** or internet TCP/IP address, and the remote application port used for retrieving the corresponding remote application. For example, the following line in the HTML application language connects a web page to the second instrument:

```
<A HREF="http://rocky13:8080">
```

For example, Uniform Resource Locator (URL) **72** may be for any of a variety of instruments such as a signal source, a signal analyzer, a meter, or a sensor. The instruments represented by URLs on web page 70 do not have to exist in one physical location. For example, one instrument could be on a manufacturing line in Japan and an another instrument could be in the United States. Alternatively, the instruments on web page 70 could be positioned at different points in a single company's manufacturing line. This would allow for remote status, diagnostics, data acquisition and more to be accessed for a variety instruments from a single web page. This could also allow for the manufacturer of an instrument to perform remote debugging of instruments without sending a field service engineer to the actual site of the instruments. In this case, each dedicated web page for an instrument will show, in addition to data, remote diagnostics such as alarms when thresholds are exceeded. As necessary, security is provided to restrict access without correct authorization codes to socket services.

After selecting instrument **20**, instrument **20** dynamically loads a web page from instrument **20** to the web browser. The web page is, for example, written in the HTML application language and has a link to automatically load the application classes to perform remote front panel control and data acquisition of remote application **10** before sending selected ASCII protocols to the dedicated socket services within instrument **20**. For example, the application classes are JAVA applet network applications.

FIG. **3** shows a web page **41** resulting when remote application **10** communicates with instrument **20** in a continuous refresh mode used for continuous data acquisition of measurement data. On a graticule **44**, two overlapped traces of scaled measurement data are shown displayed. Alternatively, four or some other number of overlapped traces of scaled measurement data are displayed.

A top scroll bar **45** provides the ability to pan the traces on the horizontal axis. Axes are interpreted differently by remote application **10** depending on the measurement chosen. For example, the horizontal axis could represent time or frequency. A left scroll bar **46** provides the ability to pan the traces on the vertical axis. For example, the vertical axis could represent amplitude. A bottom scroll bar **47** provides the ability to scale the traces up or down on the horizontal axis. A right scroll bar **48** provides the ability to scale the traces up or down on the vertical axis. A user can utilize buttons **42** to control the information flow of data between instrument **20** and remote application **10**. In the preferred embodiment of the present invention, once information is acquired by remote application **10** from instrument **20**, remote application **10** performs, as requested by a user, post processing of the acquired data. The post processing includes, for example, placement of the data in graphic form, as illustrated by the traces placed on graticule **44**. Additional post processing can include, for example, providing an average of 100 or more traces in the form of a histogram display, or any other form of post processing as limited only by the processing power of the host running remote application **10**.

A dialog box **49** shows ASCII results acquired from the results vector class of instrument **20**. A scroll bar **50** and a scroll bar **51** are used to scroll through data in dialog box **49**. A dialog box **52** shows status of remote application **10**. A scroll bar **53** and a scroll bar **54** are used to scroll through data in dialog box **52**.

Buttons **42** and **43** form a graphical user interface that is a remote front panel used to control instrument **20**. A user can utilize the graphical user interface to control the information generated by instrument **20**.

FIG. **4** shows a more sophisticated graphical user interface **60** that is a remote front panel used to control instrument **20**. Graphical user interface **60** implements advanced features such as markers and labeling. The marker feature allows a user to mark data for later reference. The labeling feature allows a user to associate labels with particular subsets of data.

FIG. **5** illustrates data acquisition from instrument **20** by the remote application **10**. Application **10** acquires data from instrument **20** using a request/response protocol. Results vector module **13**, in response to instructions from a user, sends a request **31** to server socket **23** over port **8**, which is dedicated to the transfer of results vectors. Request **31** is for example, an ASCII string. In the preferred embodiment, the ASCII string is "JAVA RV". After forwarding the request to server socket **23**, results vector module **13** waits for a results vector to be returned. After server socket **23** receives the request, instrument **20** parses the results vector request and waits for a new results vector. Measurement object **27** forwards a results vector object **34** to server socket **23** after the next measurement is made. Access to the current results vector within instrument **20** is kept under a mutually-exclusive lock. Server socket **23** then transmits a results vector **32** to remote application **10**.

In the preferred embodiment of the present invention, results vector **32** is in the form of four arrays of arbitrary length followed by an arbitrary length string which encodes ASCII results data from instrument **20**. Server socket **23** constructs results vector **32** in accordance with the algorithm set out in Table 1 below:

TABLE 1

```
for each array of data: (1 through 4)
    {   transmit number of points in array (N) (binary integer)
        for each point of data in array (1 through N)
            {   transmit binary double precision floating point
            }
    }
transmit Results Vector String Length (binary integer)
transmit Results Vector String (binary character stream)
```

In the preferred embodiment, data is compressed with run-length encoding and uses no delimiters. Each array gives information about a single trace. The information is interpreted differently depending on the measurement chosen. The results vector string is an arbitrary length null terminated string encoding any number of scalar results. The results vector also may be interpreted differently depending on the measurement chosen. For example, Table 2 below shows the format for a results vector:

TABLE 2

```
ASCII Name <sp> ASCII Value Measured <sp> ASCII Units <cr>
ASCII Name <sp> ASCII Value Measured <sp> ASCII Units <cr>
ASCII Name <sp> ASCII Value Measured <sp> ASCII Units <cr>
. . .
```

In Table 2 above, <cr>, an abbreviation for <carriage return>, is used as a delimiter between scalar results. Also, <sp>, an abbreviation for <space>, is used as delimiter between the result name and its value and between the value measured and the units.

Upon receipt of results vector **32** by results vector module **13**, remote application **10** translates the binary integers, double precision floating point values and binary characters into formats which are native to remote application **10** in order to generate a results vector **35**. After performing post processing of the acquired data within results vector **35**, remote application **10** displays the acquired data on a display **36**. For example, as shown within graticule **44** in FIG. **3**, the data is displayed in a rectangular grid plot.

The present invention allows for flexibility in accessing an instrument from a remote application. For example, using the principles of the present invention, a World Wide Web Browser with the capability of using a JAVA applet application, from any location in the world can access an instrument, such as a vector modulation analyzer (VMA) instrument, for remote control, data acquisition of measurement results, screen capture, and status, remote diagnostics and many other uses.

While particular user interfaces and transforms have been described herein, within the umbra of the present invention, new user interfaces can be designed along with new transform (new mathematical transforms, new measurements, or new hardware control/setting) functionalities (changing the resulting data acquisition). Also, since the remote application is stored in the instrument and dynamically loaded to a host this allows the remote application to be upgraded over time at the site of the instrument. This can significantly extend the life of the instrument as well as provide versatility in adapting the instruments to run with remote applications which use various programming platforms.

Further, data acquisition brought in by remote application **10** can be dynamically exported to another program such as the LabView Analysis software running in the same environment and operating system as remote application **10**. The LabView Analysis software is available from National Instruments having a business address of 6504 Bride Point Parkway, Austin, Tex. 78730-5039. The Windows operating system, for example, could serve as an environment and operating system suitable for running remote application **10** along with the LabView Analysis software. The Windows operating system is available from Microsoft Corporation having a business address of One Microsoft Way, Redmond, Wash. 98052-6399. Data acquisitions can be exported by remote application **10** to various binary and ASCII file formats, for example to a format compatible with the Excel spreadsheet application available from Microsoft Corporation, the LabView application available from National Instruments and the VEE application available from Hewlett Packard Company having a business address of 3000 Hanover Street, Palo Alto, Calif. 94304.

This versatility also allows for flexibility in providing, for example, context sensitive help retrieval, hot links (such as "Java Beans" linking utilized by the JAVA programming language) and export capability.

Once the dedicated server sockets of instrument **20** are made available over the internet, any remote application, with appropriate capability, can access instrument **20** socket services **21**, **22** ,**23** **24**. This allows drivers within existing host applications such as the LabView Analysis software available from National Instruments and the VEE application available from Hewlett Packard Company to be written in any socket networking language to enable access to the dedicated server sockets of instrument **20** for functionality, as discussed above, by remote application **10**.

The foregoing discussion discloses and describes merely exemplary methods and embodiments of the present invention. As will be understood by those familiar with the art, the invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.

I claim:

1. An instrument comprising:

storage means for storing control and data acquisition software;

server means for, upon request from a remote host system, forwarding the control and data acquisition software to the remote host system;

control response means for, in response to control commands from the control and data acquisition software running on the remote host system, controlling data acquisition of the instrument; and,

data transmission means for, in response to a data request from the control and data acquisition software running on the remote host system, forwarding acquired data from the instrument to the remote host system, wherein the data transmission means, in response to the data request from the control and data acquisition software running on the remote host system, forwards to the remote host system a results vector.

2. An instrument as in claim **1** wherein the control and data acquisition software is an application which runs as part of an internet browser.

3. An instrument as in claim **1** wherein the server means functions as a HyperText Transfer Protocol (HTTP) server and processes GET commands and PUT commands.

4. An instrument as in claim **1** wherein the control and data acquisition software includes software which, when run on the remote host system, provides a user of the remote host system a graphical user interface which the user can utilize to interact with the instrument.

5. An instrument as in claim **1** wherein the control response means is a server socket dedicated to handling messages pertaining to control of the instrument.

6. A method for providing access to an instrument from a remote host system, the method comprising the steps of:

(a) storing control and data acquisition software within the instrument;

(b) in response to the remote host system, forwarding the control and data acquisition software to the remote host system;

(c) running the control and data acquisition software on the remote host system;

(d) in response to control commands from the control and data acquisition software running on the remote host system, controlling data acquisition of the instrument; and,

(e) in response to a request from the control and data acquisition software running on the remote host system, forwarding data acquired in step (d) from the instrument to the remote host system, wherein the data acquired in step (d) is arranged in form as a results vector when sent from the instrument to the remote host system.

7. A method as in claim **6** wherein in step (c) the control and data acquisition software runs as an application within an internet browser.

8. A method as in claim **6** wherein the control and data acquisition software is forwarded to the remote host system in response to a HyperText Transfer Protocol (HTTP) server GET command.

9. A method as in claim **6** wherein step (c) includes providing a user of the remote host system a graphical user interface which the user can utilize to interact with the instrument.

10. A method as in claim **6** additionally comprising the following step:

(f) displaying on the remote host system the data forwarded in step (e).

11. A method as in claim **10** wherein step (f) includes displaying in graphic form as traces on a graticule the data forwarded in step (e).

12. A method as in claim **6** additionally comprising the following step:

processing the data forwarded in step (e) before displaying the data forwarded in step (e) on the host system.

13. Remote instrument control and data acquisition software which when run on a host computing system, comprises:

a control module which provides a user of the host computing system within a world wide web page, a graphical user interface which the user can utilize to interact with a remote instrument; and,

data acquisition means for transmitting to the remote instrument, over an internet socket, commands which control data acquisition of the remote instrument, and for receiving via internet transfer, data generated by the remote instrument, wherein the data acquisition means includes means for requesting and receiving a results vector from the remote instrument.

14. Remote instrument control and acquisition software as in claim **13** wherein the control and data acquisition software is an application which runs within an internet browser.

15. Remote instrument control and acquisition software as in claim **13** wherein the control module includes implements a network socket dedicated to handling messages pertaining to control of the remote instrument.

16. Remote instrument control and acquisition software as in claim **13** wherein the control module includes a network socket dedicated to requesting and receiving data from the remote instrument.

17. Remote instrument control and acquisition software as in claim **13** additionally comprising:

display means for displaying on the host system the data generated by the remote instrument.

18. Remote instrument control and acquisition software as in claim **17** wherein the display means displays the data in graphic form as traces on a graticule.

19. Remote instrument control and acquisition software as in claim **17** additionally comprising:

processing means which processes the data before the data is displayed by the display means.

20. A method for providing access to an instrument from a remote host system, the method comprising the steps of:

(a) storing control and data acquisition software within the instrument;

(b) in response to the remote host system, forwarding the control and data acquisition software to the remote host system;

(c) running the control and data acquisition software on the remote host system;

(d) in response to control commands from the control and data acquisition software running on the remote host system, controlling data acquisition of the instrument;

(e) in response to a request from the control and data acquisition software running on the remote host system, forwarding data acquired in step (d) from the instrument to the remote host system; and,

(f) displaying on the remote host system the data forwarded in step (e), wherein step (f) includes displaying in graphic form as traces on a graticule the data forwarded in step (e).

21. Remote instrument control and data acquisition software which when run on a host computing system, comprises:

a control module which provides a user of the host computing system within a world wide web page, a graphical user interface which the user can utilize to interact with a remote instrument;

data acquisition means for transmitting to the remote instrument in response to a request over an internet socket, commands which control data acquisition of the remote instrument, and for receiving via internet transfer, data generated by the remote instrument; and,

display means for displaying on the host system the data generated by the remote instrument, wherein the display means displays the data in graphic form as traces on a graticule.

* * * * *

US005991795A

United States Patent [19]

Howard et al.

[11] **Patent Number: 5,991,795**

[45] **Date of Patent: Nov. 23, 1999**

[54] **COMMUNICATION SYSTEM AND METHODS USING DYNAMIC EXPANSION FOR COMPUTER NETWORKS**

[75] Inventors: **Michael L. Howard**; **Christopher S. Sontag**, both of Sandy, Utah

[73] Assignee: **emWare, Inc.**, Salt Lake City, Utah

[21] Appl. No.: **08/844,158**

[22] Filed: **Apr. 18, 1997**

[51] **Int. Cl.**^{6} **G06F 13/00**

[52] **U.S. Cl.** **709/201**; 709/202; 709/217

[58] **Field of Search** 709/200, 201, 709/202, 203, 212, 213, 217, 218, 219, 225, 226, 230, 300; 707/10, 100, 104

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,287,507	2/1994	Hamilton et al.	709/303
5,442,771	8/1995	Filepp et al.	709/219
5,452,447	9/1995	Nelson et al.	707/205
5,511,208	4/1996	Boyles et al.	709/223
5,568,181	10/1996	Greenwood et al.	348/7
5,706,507	1/1998	Schloss	707/104
5,881,229	3/1999	Singh et al.	709/203

OTHER PUBLICATIONS

L. Bormann et al., "A Subsystem for Swapping and Mapped File I/O on Top of Chorus," 13th IEEE Int. Conference on Distributed Computer Systems, 1993, pp. 12–19.

A. Tomasic et al., "Caching and Database Scaling in Distributed Shared–Nothing Information Retrieval Systems," Stanford University Department of Computer Science, Report No. STAN–CS–92–1456, 1992.

S. Miller et al., "A Distributed User Information System," Institute for Advanced Computer Studies and Department of Computer Science, University of Maryland, Report No. UMIACS–TR–90–35, 1990.

E. Bertino et al., "Multos: a Document Server for Distributed Office Systems," Lecture Notes in Computer Science 303, Proceedings of the Int. Conference on Extending Database Technology, 1988, pp. 606–615.

K. Kono et al., "Smart Remote Procedure Calls: Transparent Treatment of Remote Pointers," 14th IEEE Int. Conference on Distributed Computer Systems, 1994, pp. 142–151.

"Oracle's Vision of Networked Future," by Martyn Williams, in "Trends," Newsbytes News Network, Oct. 5, 1995.

J. Ordille et al., "Distributed Active Catalogs and Meta–Data Caching in Descriptive Name Services," IEEE Int. Conference on Distributed Computer Systems, 1993, pp. 120–129.

C. Clifton et al., "Distributed Processing of Filtering Queries in HyperFile," 11th IEEE Int. Conference on Distributed Computer Systems, 1991, pp. 54–64.

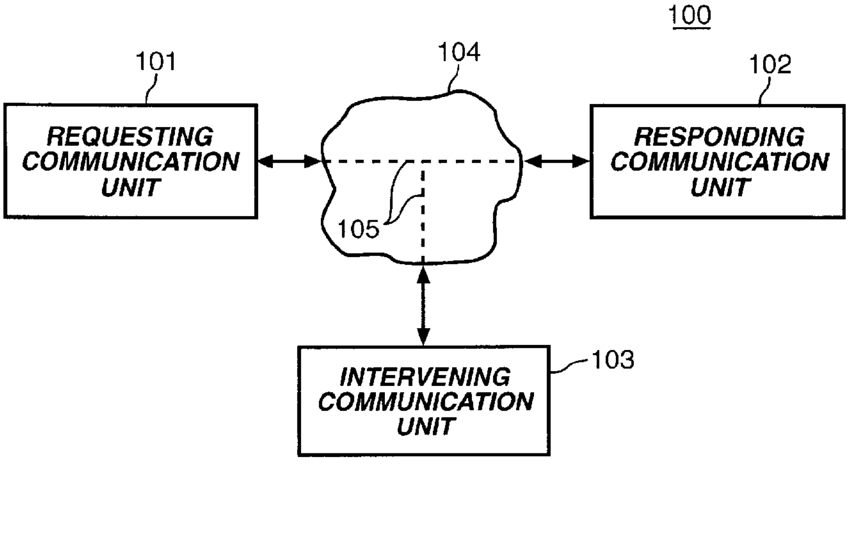
P. Malkin, "The KCM Distributed Database Management System," IBM Research Report RC 18608 (81416), Dec. 22, 1992.

Primary Examiner—Viet D. Vu
Attorney, Agent, or Firm—Parsons Behle & Latimer

[57] ABSTRACT

A method for exchange of information between a requesting communication unit and a responding communication unit through an intervening communication unit. The method comprises the steps of transmitting a request for a predetermined information element to the responding communication unit using a first format, intercepting the request for the predetermined information element, searching resources local to the intervening communication unit for the requested predetermined information element, and transmitting the requested predetermined information element to the requesting communication unit. If the requested predetermined information element is not located in local resources, the method includes the steps of translating the request for the predetermined information element into a second format and transmitting the translated request to the responding communication unit, receiving the translated request, transmitting a representation of the requested predetermined information element to the requesting communication unit, intercepting the transmitted representation of the requested predetermined information element, expanding the representation to provide the requested predetermined information element, and transmitting the requested predetermined information element to the requesting communication unit. Apparatus for exchange of information is also described.

28 Claims, 5 Drawing Sheets


100
101
104
102
REQUESTING COMMUNICATION UNIT
RESPONDING COMMUNICATION UNIT
105
INTERVENING COMMUNICATION UNIT
103

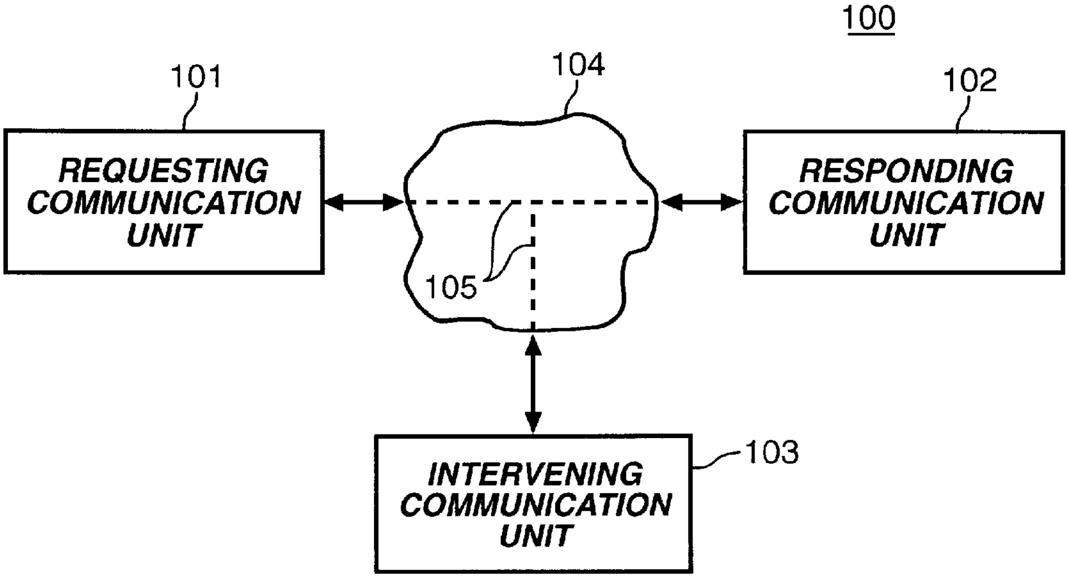

100
101
REQUESTING COMMUNICATION UNIT
104
102
RESPONDING COMMUNICATION UNIT
105
INTERVENING COMMUNICATION UNIT
103

Fig. 1

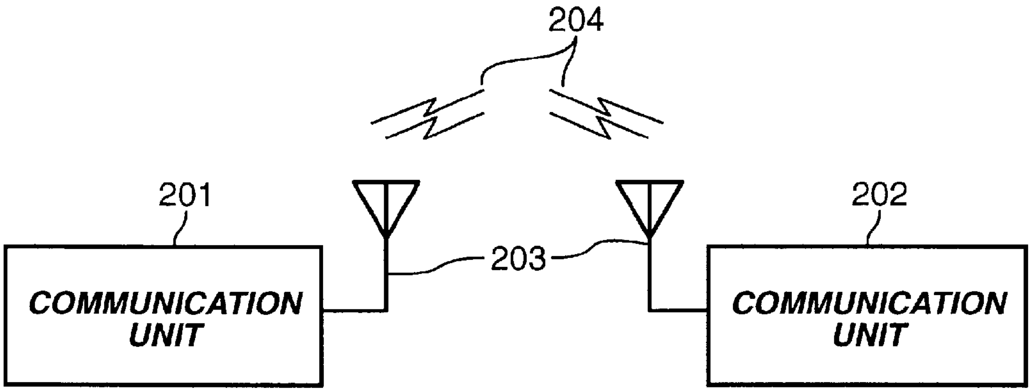

204
201
COMMUNICATION UNIT
203
202
COMMUNICATION UNIT

Fig. 2

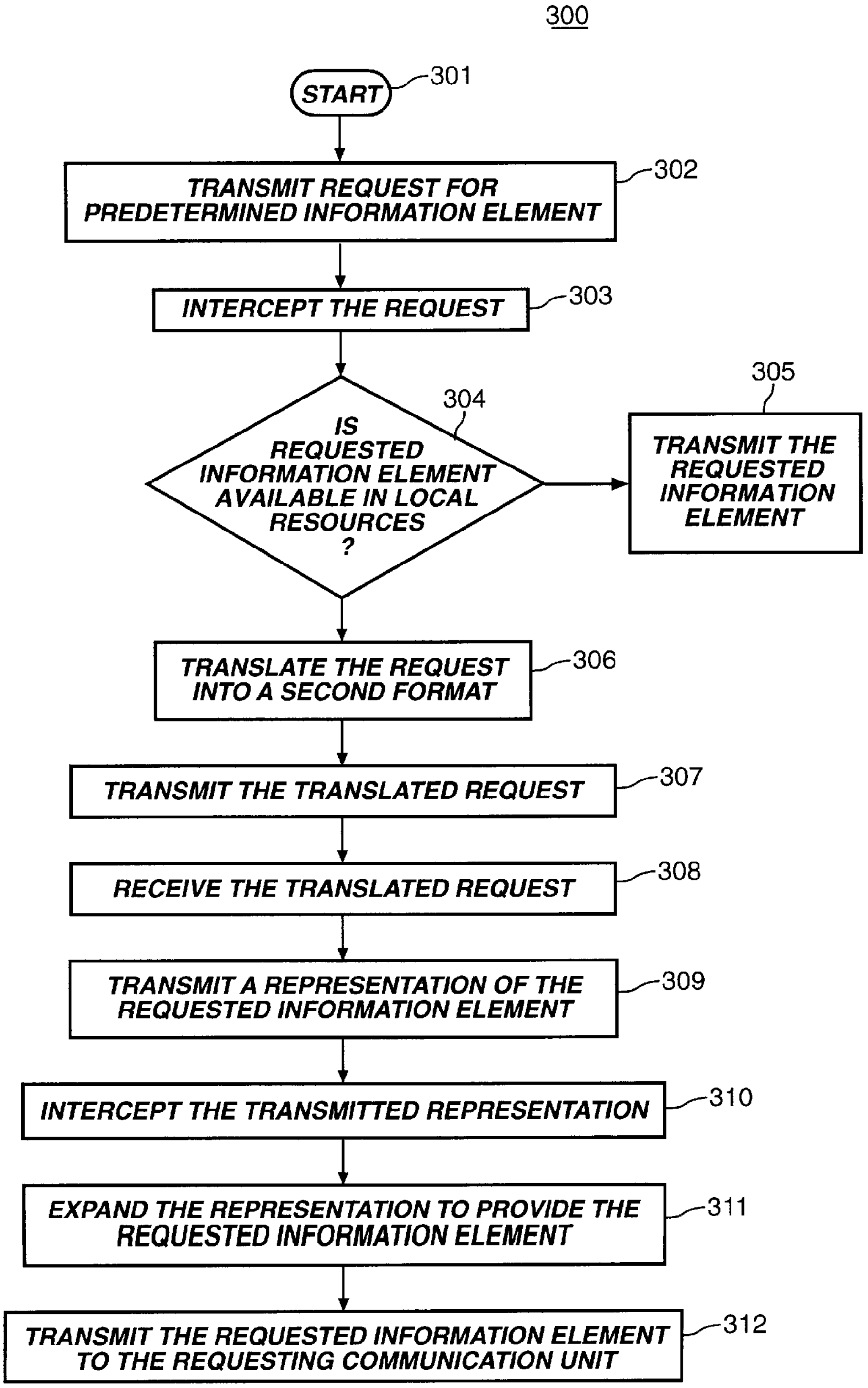

300
START
301
TRANSMIT REQUEST FOR PREDETERMINED INFORMATION ELEMENT
302
INTERCEPT THE REQUEST
303
IS REQUESTED INFORMATION ELEMENT AVAILABLE IN LOCAL RESOURCES ?
304
TRANSMIT THE REQUESTED INFORMATION ELEMENT
305
TRANSLATE THE REQUEST INTO A SECOND FORMAT
306
TRANSMIT THE TRANSLATED REQUEST
307
RECEIVE THE TRANSLATED REQUEST
308
TRANSMIT A REPRESENTATION OF THE REQUESTED INFORMATION ELEMENT
309
INTERCEPT THE TRANSMITTED REPRESENTATION
310
EXPAND THE REPRESENTATION TO PROVIDE THE REQUESTED INFORMATION ELEMENT
311
TRANSMIT THE REQUESTED INFORMATION ELEMENT TO THE REQUESTING COMMUNICATION UNIT
312

Fig. 3

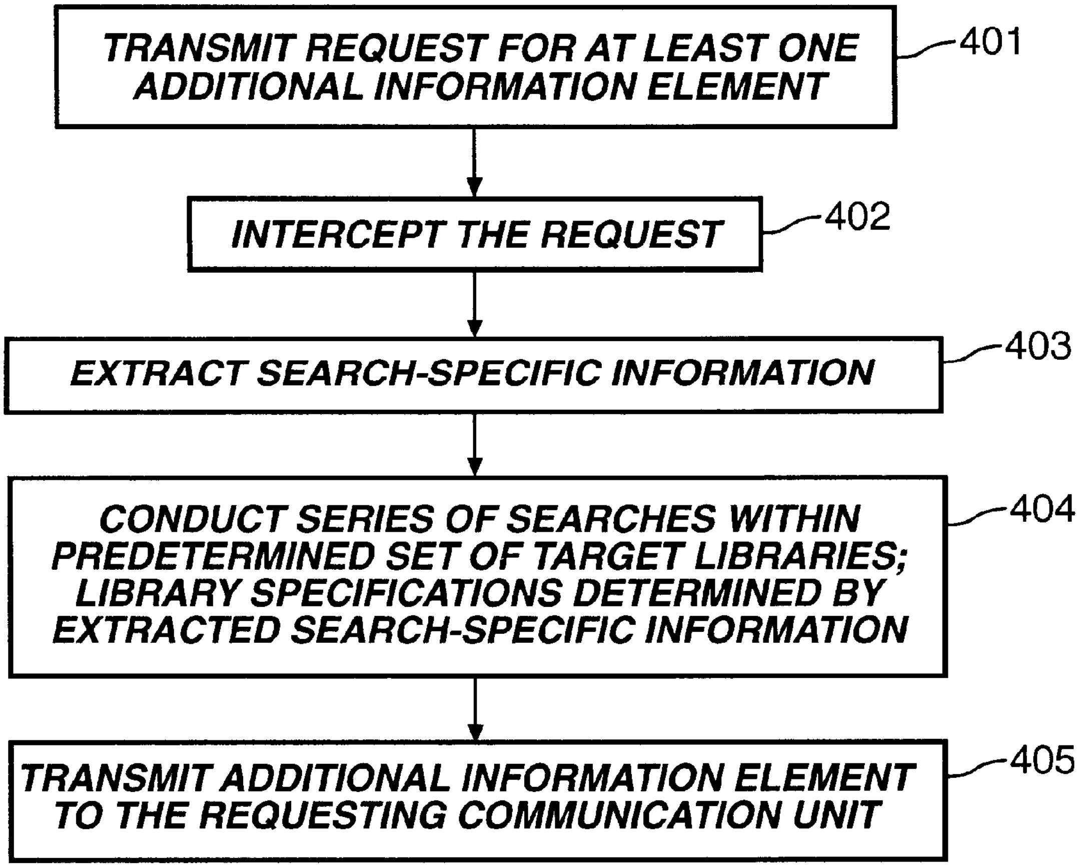

TRANSMIT REQUEST FOR AT LEAST ONE ADDITIONAL INFORMATION ELEMENT
401
INTERCEPT THE REQUEST
402
EXTRACT SEARCH-SPECIFIC INFORMATION
403
CONDUCT SERIES OF SEARCHES WITHIN PREDETERMINED SET OF TARGET LIBRARIES; LIBRARY SPECIFICATIONS DETERMINED BY EXTRACTED SEARCH-SPECIFIC INFORMATION
404
TRANSMIT ADDITIONAL INFORMATION ELEMENT TO THE REQUESTING COMMUNICATION UNIT
405

Fig. 4

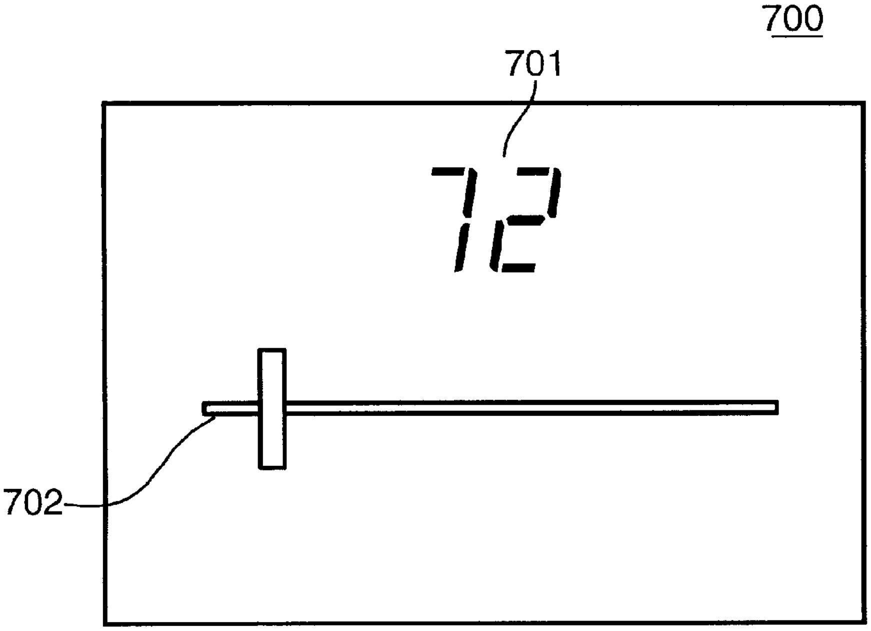

700
701
72
702

Fig. 7

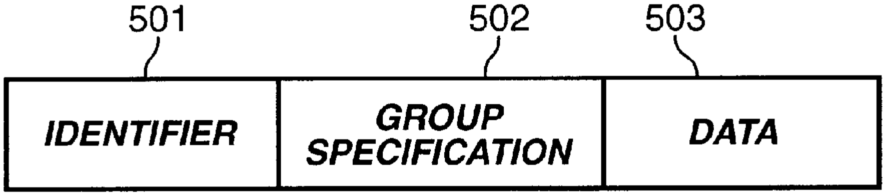

501
502
503
IDENTIFIER
GROUP SPECIFICATION
DATA

Fig. 5

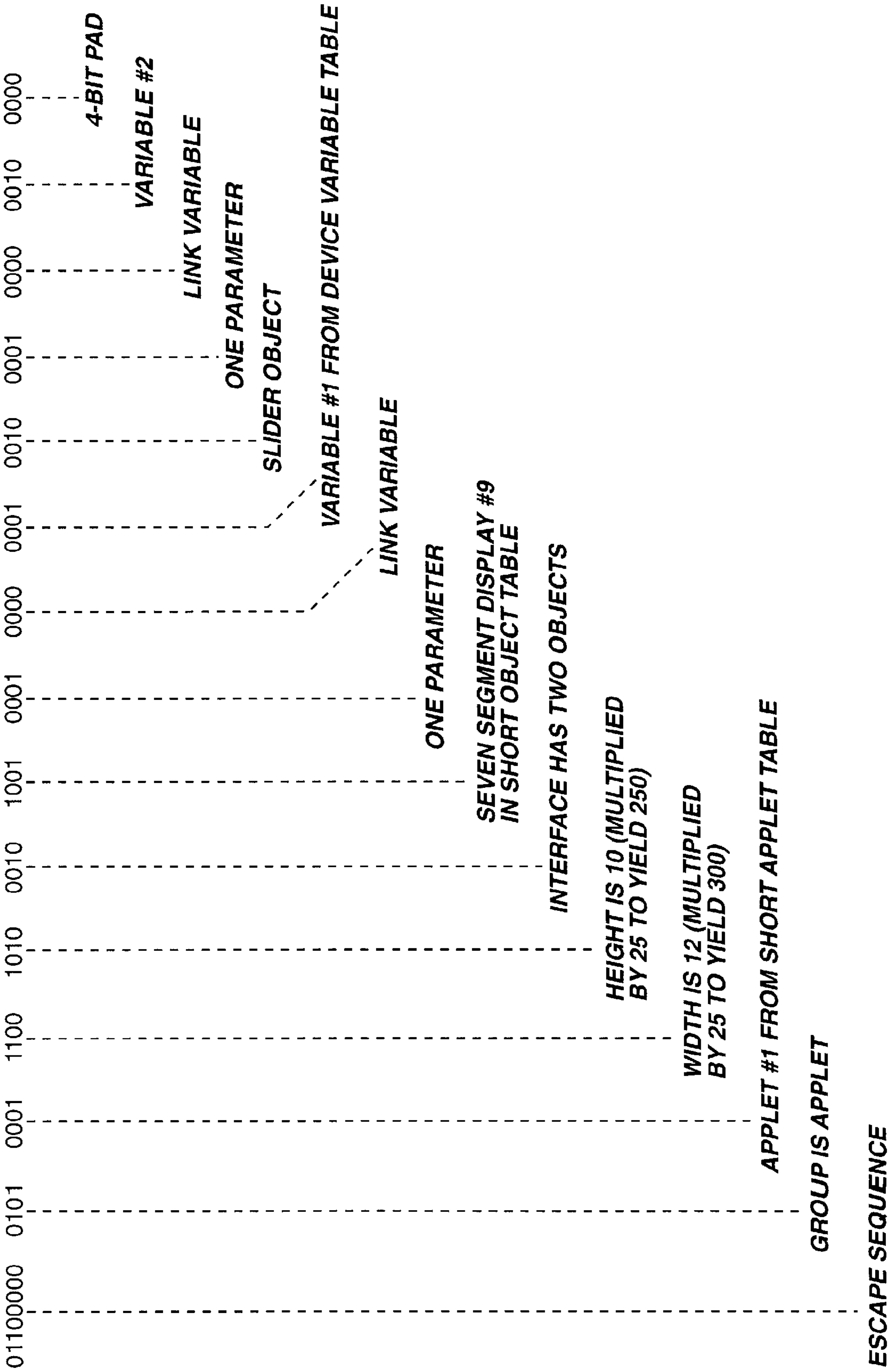

0000
4-BIT PAD
0010
VARIABLE #2
0000
LINK VARIABLE
0001
ONE PARAMETER
0010
SLIDER OBJECT
0001
VARIABLE #1 FROM DEVICE VARIABLE TABLE
0000
LINK VARIABLE
0001
ONE PARAMETER
1001
SEVEN SEGMENT DISPLAY #9
IN SHORT OBJECT TABLE
0010
INTERFACE HAS TWO OBJECTS
1010
HEIGHT IS 10 (MULTIPLIED
BY 25 TO YIELD 250)
1100
WIDTH IS 12 (MULTIPLIED
BY 25 TO YIELD 300)
0001
APPLET #1 FROM SHORT APPLET TABLE
0101
GROUP IS APPLET
01100000
ESCAPE SEQUENCE

Fig. 6

COMMUNICATION SYSTEM AND METHODS USING DYNAMIC EXPANSION FOR COMPUTER NETWORKS

FIELD OF THE INVENTION

This invention relates generally to a communication protocol and methodology useful over a computer network, and more particularly to the exchange of information with embedded systems over a computer network, and is more particularly directed toward a method and protocol aimed at minimizing overhead and memory requirements for embedded systems in communication with other computers.

BACKGROUND OF THE INVENTION

In general, the term "embedded system" connotes a combination of computer hardware and software that forms a component of some larger system. Embedded systems are intended to operate without intervention by the larger system of which they form a part, and without the need for frequent control or supervisory inputs from any source. Embedded systems are usually of simple design, and often do not include mass storage components or complex peripherals.

Embedded systems have been placed in vending machines to provide a simple computer network interface to report the need for service or vended product replenishment. Some copying machines in the office environment also utilize embedded systems for reporting of operational status, and embedded systems have been suggested in television set-top boxes designed to provide Internet web browser features through the viewer's television set.

More complex control applications represent a real challenge for embedded systems, however. Because an embedded system generally lacks large-scale storage capability, conventional methods of information interchange that are common over computer networks such as the Internet have generally been eschewed in favor of much simpler protocols.

But the communication capability of the web browser-web server interconnection promises great flexibility for control applications. Graphics-rich web pages with embedded hypertext links offer intuitive status and control environments that are attractive to the designers of embedded control applications.

Caching systems have been suggested as one way to avoid burdening the limited storage capability of embedded systems. In a caching system, a data entity is stored on a system, retrieved, then saved on the referencing device in a cache in order to reduce system overhead in response to subsequent references. However, caching cannot deal with a scenario in which the referenced object has never been stored. Due largely to this limitation, caching is not generally considered appropriate in command and control applications.

Libraries of objects for which a need is anticipated can overcome some of the onerous storage requirements of large objects. But libraries suffer from at least two shortcomings that make their use with embedded systems less attractive.

First, libraries can only store specific objects for which a need is anticipated. A library cannot dynamically support a request for objects of a certain type, for example, rather than a request for an object of a certain name. And, if the requested object type is not located in the library, the library cannot dynamically conduct an ordered search of other libraries in an effort to locate instances of the requested object type, since the order would vary with the type of requested object.

Second, a library cannot store sets of commonly used parameter values, nor can a library make decisions concerning default parameter values, or the best values to use for a given situation described by another set of parameters.

Accordingly, a need arises for an information interchange capability aimed at embedded systems that allows these streamlined microcomputers to implement status and control applications using the graphics capabilities of web browser programs over the Internet and World Wide Web, as well as over other types of computer networks.

SUMMARY OF THE INVENTION

These needs and others are satisfied by the method of the present invention for exchange of information between a requesting communication unit and a responding communication unit through an intervening communication unit. The method comprises the steps of, at the requesting communication unit, transmitting a request for a predetermined information element to the responding communication unit using a first format, at the intervening communication unit, intercepting the request for the predetermined information element, searching resources local to the intervening communication unit for the requested predetermined information element, and, if the predetermined information element is located in local resources, transmitting the requested predetermined information element to the requesting communication unit. Else, if the requested predetermined information element is not located in local resources, the method includes the steps of translating the request for the predetermined information element into a second format and transmitting the translated request to the responding communication unit. At the responding communication unit, additional steps of the method include receiving the translated request, transmitting a representation of the requested predetermined information element to the requesting communication unit, and at the intervening communication unit, intercepting the transmitted representation of the requested predetermined information element, expanding the representation to provide the requested predetermined information element, and transmitting the requested predetermined information element to the requesting communication unit.

The step of transmitting a request for a predetermined information element comprises the step of requesting a predetermined document via HTTP. The step of translating the request for the predetermined information element into a second format comprises the step of translating the request into a protocol that is an enhanced version of HTTP.

In one aspect of the invention, the step of transmitting a representation of the requested predetermined information element comprises transmitting a compressed representation of the requested information element including at least one abridged representation of an object associated with the requested information element. The object associated with the requested information element may comprise an executable program element or an image file, for example.

In another aspect of the invention, the step of expanding the representation comprises the step of dynamically expanding the representation by acquiring referenced objects associated with the requested information element modeled by the representation.

In accordance with one form of the invention, the predetermined information element specifies at least one additional information element, and the method further comprises the steps of, at the requesting communication unit, transmitting a request for the at least one additional information element to the responding communication unit, and, at the intervening communication unit, intercepting the request for the at least one additional information element, extracting search-specific information regarding the at least one additional information element from the intercepted request, conducting a series of searches for the at least one additional information element within a predetermined set of target libraries, wherein library search chronology and specific target library designation are determined by the extracted search-specific information, and, transmitting the at least one additional information element to the requesting communication unit.

The step of transmitting a request for the at least one additional information element comprises the step of transmitting a request for at least one additional document referenced in the originally requested predetermined information element, and the step of extracting search specific information comprises the step of examining document class designations for the additional documents referenced.

In another form of the invention, the step of conducting a series of searches further comprises the steps of, first, searching document libraries on local storage media with order of search determined at least in part by document class designation, and subsequently searching document libraries associated with remote communication units.

In another aspect of the invention, network apparatus for exchange of information comprises means for transmitting a request for a predetermined information element from a requesting communication unit to a responding communication unit using a first format, means for intercepting the request for the predetermined information element at an intervening communication unit, means for searching resources local to the intervening communication unit for the requested predetermined information element, means for determining whether the requested predetermined information element is located in local resources, means for transmitting the requested predetermined information element to the requesting communication unit if the requested predetermined information element is located in local resources, means for translating the request for the predetermined information element into a second format provided that the requested predetermined information element is not located in local resources, means for transmitting the translated request to the responding communication unit, means for receiving the translated request at the responding communication unit, means for transmitting a representation of the requested predetermined information element to the requesting communication unit, means for intercepting the transmitted representation of the requested predetermined information element at the intervening communication unit, means for expanding the representation to provide the requested predetermined information element, and means for transmitting the requested predetermined information element to the requesting communication unit.

The means for transmitting a request preferably comprises a requesting computer system interconnected with the responding communication unit. The requesting computer system may include web browser software. The means for intercepting the request preferably comprises an intervening communication unit interconnected with the requesting communication unit. The intervening communication unit may be a communication unit selected from the group of communication units consisting of a file server, a proxy server, and a common gateway interface.

In yet another aspect of the present invention, a method for implementing an abridged communication protocol for exchange of information among communication units, wherein a designated server communication unit includes, accessible in local storage, a document directory having a symbolic reference to available documents, selected parameter values, and selected operations comprises the steps of, at a requesting communication unit, transmitting a request to the designated server communication unit for a predetermined document defining selected protocol standards, and upon receiving the predetermined document, formatting a request in accordance with the protocol standards, and transmitting the request to the designated server communication unit. The method further includes the steps, at the designated server communication unit, of receiving the transmitted request, extracting request-specific information from the received request, and, if the request is for a document, using a received document identifier as an index into the symbolic reference to available documents, accessing and transmitting the requested document to the requesting communication unit. If the request is for a parameter value, further steps include using a received parameter identifier as an index into the symbolic reference to selected parameter values, accessing and transmitting the requested parameter to the requesting communication unit, and, if the request is for an operation, using a received operation identifier as an index into the symbolic reference to selected operations, performing the selected operation.

The abridged communication protocol may comprise an enhanced version of HTTP. The step of transmitting a request to the designated server communication unit comprises transmitting a request for a capabilities document that includes indicia representing software version and revision level, ordering of data elements, and size of data elements. The symbolic reference to available documents comprises a document directory having a symbol table including entries corresponding to documents, parameter values and operations stored on or recognized by the server communication unit. Documents stored on or recognized by the server communication unit are of a type selected from the set of document types consisting of: static documents, dynamic documents, invocation documents, and variable manipulation documents.

Static documents comprise documents that change only when software version or revision level changes, dynamic documents comprise documents that change from time to time, and each dynamic document includes a stored modification date and a stored revision date, an invocation document invokes an operation, and a variable manipulation document manipulates a parameter value.

In yet another aspect of the invention, the capabilities document is the first ordered document stored on the server and is denominated document **0**. The document directory is the second ordered document stored on the server and is denominated document **1**. The symbolic reference to selected parameter values is the third ordered document stored on the server and is denominated document **2**.

In still another form, a method for providing an abridged reference to information elements exchanged among communication units by representing a selected information element as an ordered sequence of fields, comprises the steps of providing a first field identifying subsequent fields as part of an abridged representation of an information element, providing a second subsequent field identifying a type of information element represented, and providing a third subsequent field identifying a library and an index value into a symbolic representation of library contents.

The step of providing a first field identifying subsequent fields as part of an abridged representation of an information element further comprises the step of providing a first field comprising a predetermined bit pattern. The predetermined bit pattern is selected based upon rarity of occurrence in normal text.

The step of providing a second subsequent field identifying a type of information element represented further comprises the step of identifying a type of information element selected from the group consisting of executable files, image files, rich text files, and configuration files. Executable files may comprise Java applets, and rich text files may comprise files conforming to a mark-up language specification. The mark-up language may comprise HTML.

In yet another aspect of the invention, the step of providing a third subsequent field identifying a library and an index value into a symbolic representation of library contents further comprises the step of providing a first subfield that includes a symbolic library identifier and a second subfield that includes an index value. The index value comprises an integer.

Further objects, features, and advantages of the present invention will become apparent from the following description and drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** depicts a system of communication units in block diagram form;

FIG. **2** illustrates communication units exchanging information signals via an RF interconnection;

FIG. **3** is a flow chart illustrating a communication protocol;

FIG. **4** is a flow chart illustrating a portion of a communication protocol;

FIG. **5** illustrates the general form of a compressed representation format;

FIG. **6** is illustrates another example of an abridged representation format; and

FIG. **7** depicts a control panel.

DETAILED DESCRIPTION OF THE INVENTION

In accordance with the present invention, a protocol and methodology for information interchange for embedded systems communicating over computer networks is described that provide distinct advantages when compared to those of the prior art.

The protocol and methodology described are designed to enable communication between an embedded system and a second computer system with which it is interconnected. However, the protocols and methodologies described are not limited to communication with embedded systems, and may be applied to any interchange of information between data communication units no matter how their interconnection is accomplished.

In one embodiment of the invention, information interchange is described between an embedded system that acts as a web server, and a web browser of conventional design, to achieve the same graphics-rich, robust interface that could be obtained with a conventional web server. In general, the methodology described may be implemented among communication units that are interconnected for the purpose of exchanging information. Such a system of communication units is depicted in block diagram form in FIG. **1**.

A communication system, generally depicted by the numeral **100**, includes a number of communication units **101–103** that are interconnected via an advantageous interface **104**. Each of the communication units **101–103** may be a host computer system, a personal computer, a file server, a proxy server, a computer device acting as a common gateway interface (CGI), or a terminal device having varying degrees of "intelligence" as known in the art. Of course, the specific type of communication unit is not limited to the types of devices enumerated here. Each of the communication units **101–103** has the capability of exchanging information with other communication units.

The specific type of interface or communication channel **104** is not limited to a local area network (LAN) as known in the art. The communication channel **104** may include a wide area network (WAN) as one of its elements, or interconnection among communication units **101--103** may be accomplished in part via the network of computer networks known as the Internet. FIG. **2** illustrates yet another communication channel that may form at least a part of the interface **104** depicted in FIG. **1**.

In FIG. **2**, communication units **201**, **202** are illustrated exchanging information signals **204** via a radio frequency (RF) interconnection through appropriate antenna systems **203** that are suitable for the frequency of interest. This RF communication channel may be implemented via known RF modem technology, with an RF modem replacing the conventional telephone modem often used for intercommunication between computer systems.

Of course, implementation of an RF communication channel is not limited to this suggested combination, and may also include, for example, in whole or in part, point-to-point terrestrial microwave links or satellite links, or hard-wired communication channels provided by coaxial, twisted-pair, or even fiber-optic cable systems. All or part of the interface **104** depicted in FIG. **1** may even be a direct hard-wired connection, as suggested by the dashed lines **105**.

One of the communication units **101** may be termed a requesting communication unit when it transmits a request for information to a responding communication unit **102**. A third communication unit **103** may be designated an intervening communication unit if it is designed to intercept and potentially reprocess requests for information received from a requesting communication unit **101**.

A protocol for the type of communication described above is illustrated in the flow chart of FIG. **3**, and generally depicted by the numeral **300**. After exiting a START state **301**, the requesting communication unit **101** transmits a request for a predetermined information element to the responding communication unit **102** using a first format (step **302**).

In the subsequent step **303**, the intervening communication unit **103** intercepts the request for the predetermined information element, and, in step **304**, searches resources local to the intervening communication unit **103** for the requested information element. These resources may include hard disk storage, random access memory (RAM), read-only memory (ROM) , "flash" memory, masked ROM, or various types of archival storage, such as off-line removable disk media or tape drives, for example.

If the requested information element can be located in local resources, the intervening communication unit **103** transmits the requested information element to the requesting communication unit **101** in step **305**. If, on the other hand, the requested information element cannot be located, the intervening communication unit **103** translates the request for the predetermined information element into a second format in step **306**, and in the subsequent step **(307)**, transmits the translated request to the responding communication unit **102**.

The responding communication unit **102** receives the translated request in step **308**, and transmits a representation of the requested predetermined information element to the requesting communication unit in the next step **(309)**. In the subsequent operation **310**, the intervening communication unit **103** intercepts the transmitted representation of the requested predetermined information element, expands the representation (step **311**) to provide the requested predetermined information element, and, in the subsequent operation **(312)** , transmits the requested predetermined information element to the requesting communication unit **101**.

The first format in which a transmitted request for information is set may be conventional HTTP, the hypertext transfer protocol developed by Tim Berners-Lee and others at CERN, the European Particle Physics Laboratory, and well-known to those skilled in the art. Of course, the first format is not restricted to HTTP, and may comprise any format suitable for information interchange, including proprietary formats.

The second format into which the intervening communication unit **103** (which may be a proxy server, for example, that knows the network address or other identifier of the responding communication unit) translates the request is preferably an enhanced version of HTTP. This enhanced HTTP isa specialized version of HTTP that relies on a gateway proxy to simplify HTTP and CGI operations at the responding communication unit **102**. This is particularly useful when the responding communication unit **102** is an embedded system with limited memory and processing power. In general terms, of course, this second format may be any enhanced version of the first format used.

Enhanced HTTP reduces memory and processing requirements of an embedded web server to a point where small micro-controllers can act as servers using as little as **4**k bytes (kilo-bytes) of RAM for program and data storage. As far as run-time performance is concerned, traffic between a web server on the World Wide Web and the intervening communication unit or proxy gateway **103** can be held as closely as possible to purely state and control information.

This traffic reduction results in good performance of a responding communication unit **102** even when used with very slow communication channels. Some of the desirable characteristics of this second format are:

1. The intervening communication unit **103** (proxy) is generally responsible for validation of requests from the requesting communication unit **101** and handling denial of requests where appropriate;

2. Requested information elements such as documents in the intervening communication unit (server) are accessed through an index rather than by name;

3. The commands of the second format directly support function execution and scalar variable manipulation;

4. The intervening communication unit (server), at startup, specifies byte ordering. The intervening communication unit (proxy) is always responsible for any needed translation; and

5. Special handling of variable state updates may be provided.

If the requesting communication unit **101** requires standard HTTP, the intervening communication unit (gateway) can translate enhanced HTTP into standard HTTP and act as a proxy server. Preferably, this translation capability exists regardless of the specific format chosen to be the first format or second format.

A document directory in the form of a symbol table of documents and data functions are stored at the responding communication unit **102** (server) but the server **102** does not have any logic to manipulate the contents of the tables. Instead, the server **102** accesses its documents using a document index. A document stored at a predefined index contains a compressed symbol table that is used by the intervening communication unit **103** to provide access to the documents using symbolic names.

When the requesting communication unit **101** requests a document by name, the intervening communication unit **103** verifies that the document exists, then translates the name to a document number (index). The responding communication unit **102** simply needs to use the number to get the document without needing to handle a case where the document does not exist. Having the intervening communication unit (proxy) **103** handle the name lookup and associated error handling code results in a significant reduction in programming logic requirements at the responding communication unit **102** and is essentially transparent to the requesting communication unit **101**.

The familiar HTTP GET operation, for instance, is encoded as:

```
G1 (Get document 1)
```

The capital ‘G’ stands for the enhanced HTTP syntax for GET. The number following the ‘G’ is a binary encoded unsigned short value that denotes the document number. This number is verified against the symbol table downloaded at connection initialization time. If a file is to be written, then the symbol table must of course have a predefined placeholder.

Documents may be classified into categories:

Static—Only change with product or firmware revision;

Dynamic—May change during system operation. Has a modification and creation date;

Invocation—A function is invoked, not really a document retrieval; and

Variable Manipulation—A variable is manipulated.

Static documents are loaded as needed, but may be cached in a fashion retrievable by the intervening communication unit **103**. A cached static document is associated with the product and firmware revision of the device. If these change, then the static document cache must be cleared and reloaded.

A capabilities document is defined as the first **(0)** document for systems implementing enhanced HTTP as the second format into which requests for information elements are translated by the intervening communication unit **103**. This document will define revisions, byte ordering and word size. The byte ordering specification is used by the intervening communication unit **103** (proxy) to translate all data larger than a byte into the ordering native to the responding communication unit or server **102**. The word size is used to set the size of integers used in the enhanced HTTP command set.

The enhanced HTTP data tables are as follows:

Static Document Table - These documents do not change.
Document 0: Capabilities document:

-continued

UINT8	major revision
UINT8	minor revision
UINT8	byte ordering type (this is an enumeration of all potential byte ordering variations)
The byte ordering type is used to order the following:	
UINT16	Max Message Size
UINT32	Manufacturer ID
UINT32	Product ID
UINT32	Product Revision
UINT32	Firmware Revision
Document 1:	Symbol table of all documents on the server
UINT8	Document type (1 or 2)
SYMBOL TABLE	
Document 2:	Variable Table of all variables accessible through server
UINT8	Document type (1 or 2)
VARIABLE TABLE	

User Documents are numbered **5** and higher.

Variable manipulation and function invocation are available using the following set of data types:

UINT8
INTO

UINT8ARRAY
INT8ARRAY
UINT16
INT16
UINT16ARRAY
INT16ARRAY
UINT32
INT32
UINT32ARRAY
INT32ARRAY

Array encoding [UINT16 length][data]

Length and data are stored in server's native byte order.

The following is a description of enhanced HTTP commands. The command codes are spelled out in words, but are preferably assigned eight-bit constant values. Since the gateway process described above will validate data types and symbol names, there are no response codes provided for simple function invocation or setting a variable's value.

Code	Description	Response
[GVER]	Get enhanced HTTP version	[RESP][UINT8 version]
[GIDXSZ]	Get Index Size	[RESP][UINT8]
[GSD]	Get Static Document [GSD][Index][OFFSET] Index is an index into the static document table. Gateway proxy will have already validated the entry.	[RESP][Contents]
[CALL]	Call a function This directly invokes a function in the function table. [CALL][Index] Index is a valid index into the function table.	NONE
[CALLP]	Call a function with parameters. [CALLP][Index][parameter1][parameter2] A function parameter list may be specified in the symbol table by extending the name field as follows: Name?[p1][p2][p3] . . . [pN]?[r] where "p" and "r" are UINT8 specifying the native type of the parameter/return.	VALUE
[SVXXX]	Set Variable [SVXXX][ID][Index][VAL] XXX One of the supported native data types. ID An index into the variable table. Index An index into the function table	NONE
[GVXXX]	Get Variable [GVXXX][ID][Index] XXX One of the supported native data types ID An index into the variable table. Index An index into the function table.	VALUE
[GVMXXX]	Get Variable if modified [GVMXXX][ID][Index] XXX One of the supported native data types. ID An index into the variable table. Index An index into the function table. This returns the variable if it was modified since the last unconditional get or successful call to GVM.	[No Change] or [Change][VAL]
[POST]	Post Document [POST][Index][U32 Length][DATA] Post a new file to the embedded server Post is allowed only if the document specified by [Index] has D0 & D1 of the flags set to zero.	[OK] or [ERROR][VAL] VAL: 0 - Full 1 - not Allowed

The SV/GV variable, CALL, and POST commands are only present in the server **102** if the symbol tables contain data that may be manipulated through these commands. If, for instance, there are no arrays of data in the embedded application, then the SVINTARRAY type of commands would not be sent from the intervening communication unit **103**, so including the code on the responding communication unit (server) **102** to support this would be a waste of space.

The following document types are supported:

1	Small Symbol Table (8 bit chars) UINT8 - Table Size Entries: Null Terminated string - Name of document UINT8 document number UINT8 flags: D0 1 = static, 0 = dynamic D1 1 = final, 0 = Can be written D2–D7 reserved
2	Large Symbol Table (8 bit chars) UINT16 - Table Size Entries: Null Terminated string - Name of document UINT8 document number UINT8 flags: D0 1 = static, 0 = dynamic D1 1 = final, 0 = Can be written D2–D7 reserved
3	Small Symbol Table (16 bit chars) UINT8 - Table Size Entries: Null Terminated string - Name of document UINT8 document number UINT8 flags: D0 1 = static, 0 = dynamic D1 1 = final, 0 = Can be written D2–D7 reserved
4	Large Symbol Table (16 bit chars) UINT16 - Table Size Entries: Null Terminated string - Name of document UINT16 document number UINT8 flags: D0 1 = static, 0 = dynamic D1 1 = final, 0 = Can be written D2–D7 reserved
10	Small Variable Table Null Terminated string UINT8 - variable type UINT8 - Maximum element count (1 for scalar) UINT8 - SET FUNCTION NUMBER (0 signifies simple write) UINT8 - GET FUNCTION NUMBER (1 signifies simple read)
11	Large Variable Table Null Terminated string UNSIGNED byte variable type UINT16 - Maximum element count (1 for scalar) UINT16 - SET FUNCTION NUMBER (0 signifies simple write) UINT16 - GET FUNCTION NUMBER (1 signifies simple read)
20	Enhanced HTML (The markup language of enhanced HTTP)
21	HTML
30	GIF (Graphics Interchange Format)
31	JPEG (Joint Photographic Experts Group)

Of course, the document types illustrated are not intended to exclude other types. The types illustrated merely represent preferred document types.

As discussed previously, the responding communication unit **102** may transmit only a representation of the requested information element. In order to provide an efficient method to specify documents and user interface components inside a micro-controller, a compressed representation format has been devised that may also accommodate references to abridged representations of objects associated with the originally requested information element or document. This compressed representation format is a specification of a file or Java applet that is encoded to save space in the responding communication unit. This compressed representation format also provides a more flexible means of addressing documents than simply by name.

A responding communication unit **102** that is an embedded device may reference HTML, image, or Java class files that reside on a host workstation. These components are generally not specific to a device. Instead, a toolbox of device control, data collection, event handling, and user interface components are preferably made available for the embedded device to use and link to variables or procedures on the device. The purpose of the compressed representation format is to allow specification of these components in a way that minimizes demands on memory in the embedded system. Since the compressed representation format results in abbreviated data elements being transmitted, the demand on interface bandwidth is also minimized.

The process of expanding a compressed representation format to a full HTML reference such as <APPLET . . . > or <HREF . . . > is termed dynamic expansion. The intervening communication unit **103**, whether a proxy gateway or file server, expands the compressed representation format and is also responsible for supplying any referenced files.

For a typical World Wide Web browser acting as a requesting communication unit **101**, an embedded system as the responding communication unit **102**, and a proxy gateway as the intervening communication unit **103**, the document retrieval process may be in accordance with the following:

First, the web browser initiates an HTTP GET request. The gateway receives the request, reformats it into enhanced HTTP, and forwards it to the device. The device sends the compressed document. The gateway then decompresses the document, and expands the compressed representation format.

Next, the browser reads the document and does a GET for any needed documents, such as Java class files, that were referenced in the document. The gateway receives the GET request, but looks up and sends the file from its local storage.

A compressed representation format may be placed in a document which is stored in an embedded device. When the document is retrieved from the embedded device, the communications gateway looks for a sequence that identifies the compressed representation format (escape sequence), then reads the group number following the escape sequence.

The group number specifies what type of information element the sequence represents. Using the format specified by the specific group, the compressed representation format is decoded, then expanded into its full textual representation. This full (dynamically expanded) representation is then placed into the document by the gateway before forwarding the document to the software module which originated the document retrieval (the requesting communication unit).

The general form of a compressed representation format (or abridged reference to information elements) is set forth in FIG. **5**. The compressed representation format includes an identifier portion **501**, a group specification **502**, and a data portion **503**.

The identifier portion **501** is preferably an escape sequence; that is, a sequence of bits that does not frequently appear in document text. The value of the sequence is repeated twice to represent the sequence when it appears in document text outside of the compressed representation context. The back quote character "‘" may be used to refer to this flag, but there is no reason that the escape sequence must be an eight bit value. For convenience of encoding, however, all compressed representation sequences are preferably padded to be a multiple of the number of bits in a character. Since the document may be composed of either eight or sixteen bit characters, the padding may add as much as fifteen bits.

The group specification **502** is used to differentiate which of several types of references this compressed representation (or abridged reference) should expand into. Examples are application specific Java Applet, image file, HTML file, and abridged reference configuration file. Each of these types preferably has a different encoding, which may be represented in the third (data) field **503**. The simplest are the image and HTML file formats which preferably look like the following:

```
‘i2
```

Where:

‘—Represents the escape

i—Represents the main image library

2—Specifies the second image in that library.

A practical use of such an abridged representation might be to place a large Icon in a document as a header for a help section. An icon in the image library could be placed into an HTML file using only one or two bytes, for example. Like the escape sequence, the ‘i’ and the ‘**2**’ in the tag may be stored in units less than eight bits. The sizes of these fields are partially specified in the document itself. In very memory-constrained systems, the address portions of the abridged representations are reduced in size. In larger systems, the address portions may be made larger.

This abridged representation may be viewed simply as a group of numbers. The image specification is only two numbers; the group number and the image number. Memory constrained devices may be set up to choose from only one of sixteen available groups, and only one of sixteen available images. Thus, in binary, the abridged representation may look like the following:

```
01100000 0101 0010
```

The sequence 01100000 is the designated escape sequence, 0101 identifies group number **5** (the image group), and 0010 identifies image number **2**.

A more complex example of an abridged representation format is set forth in FIG. **6**. The representation shown encodes the following information:

```
<APPLET code=emApContainer.class width=300 height=250>
<PARAM name=“OBJECTS” value=“2”>
<PARAM name=“OBJECT0” value=
“emJava.emDISPLAYS.em7SegDsp”>
<PARAM name=“OBJECT0_JRI”value=“value_temp”>
<PARAM name=“OBJECT1” value=“emJava.emSliders.emHzSlider”>
<PARAM name=“OBJECT1_JR1”value=“value_intensity”>
```

Total Length 300 bytes

This applet specification sets up a control panel **700** as illustrated in FIG. **7** that displays a temperature **701** and has a slider control **702** to control light intensity in a demonstration circuit (not shown).

Unlike HREFs and APPLET tags, the abridged representation described can refer to a document type, not just a specific document. If an applet control for a slider is chosen, for example, a request for a specific class will be made to the responding communication unit. The response to the request will be the class file that most closely matches the device being controlled. A default set of Java controls, for example, may be supplied to a user, but a set of specific controls may also be supplied for all projected applications as well as additional sets aimed at specific products.

The abridged representation may directly reference a specific control, or may reference the control for the device. In the latter ease, the intervening communication unit will attempt to retrieve the device control, but will fall back to the set of specific controls, or to the generic set of Java controls if necessary.

The abridged representation described applies to any document type, not just Java class files. A help file, for instance, may give general control usage help for a set of Java class files, but specific device help and contact links for the specific device control.

In addition to referencing a document, the abridged representation may encode most frequently used parameter values. Since the abridged representation is an enumeration, there is no restriction on the interpretation of any given field by the server. The most frequently needed fields are encoded in the first positions, so extremely memory sensitive applications can run using a minimum of document size.

FIG. **4** represents, in flow chart form, a portion of a protocol that addresses the situation in which the predetermined information element (document) returned to the requesting communication unit includes a specification of at least one other information element to be retrieved. In step **401**, the requesting communication unit transmits a request for the additional information element specified in the originally requested predetermined information element. In the subsequent step (**402**), the intervening communication unit or proxy **103** intercepts the request for the additional information element, and extracts (step **403**) search-specific information regarding the additional information element.

The proxy **103** searches for the additional information element within a predetermined set of target libraries, with specific library designations, search order, and other search parameters based upon the extracted search-specific information (step **405**). The proxy then transmits the additional information element to the requesting communication unit in the subsequent step (**405**).

There have been described herein a protocol and methodology for information interchange for embedded systems communicating over computer networks that are relatively free from the shortcomings of the prior art. It will be apparent to those skilled in the art that modifications may be made without departing from the spirit and scope of the invention. Accordingly, it is not intended that the invention be limited except as may be necessary in view of the appended claims.

What is claimed is:

1. A method for exchange of information between a requesting communication unit and a responding communication unit through an intervening communication unit, the method comprising the steps of:

at the requesting communication unit:

(a) transmitting a request for a predetermined information element to the responding communication unit using a first format;

at the intervening communication unit:

(b) intercepting the request for the predetermined information element;

(c) searching resources local to the intervening communication unit for the requested predetermined information element; and, if the predetermined information element is located in local resources,

(d) transmitting the requested predetermined information element to the requesting communication unit; else, if the requested predetermined information element is not located in local resources,

(e) translating the request for the predetermined information element into a second format; and

(f) transmitting the translated request to the responding communication unit;

at the responding communication unit:

(g) receiving the translated request;

(h) transmitting a representation of the requested predetermined information element to the requesting communication unit;

at the intervening communication unit:

(i) intercepting the transmitted representation of the requested predetermined information element;

(j) expanding the representation to provide the requested predetermined information element; and

(k) transmitting the requested predetermined information element to the requesting communication unit.

2. The method in accordance with claim **1**, wherein the step (a) of transmitting a request for a predetermined information element comprises the step of requesting a predetermined document via HTTP.

3. The method in accordance with claim **1**, wherein the step (e) of translating the request for the predetermined information element into a second format comprises the step of translating the request into a protocol that is an enhanced version of HTTP.

4. The method in accordance with claim **1**, wherein the step (h) of transmitting a representation of the requested predetermined information element comprises transmitting a compressed representation of the requested information element including at least one abridged representation of an object associated with the requested information element.

5. The method in accordance with claim **4**, wherein said object associated with the requested information element comprises an executable program element.

6. The method in accordance with claim **4**, wherein said object associated with the requested information element comprises an image file.

7. The method in accordance with claim **1**, wherein the step (j) of expanding the representation comprises the step of dynamically expanding the representation by acquiring referenced objects associated with the requested information element modeled by the representation.

8. The method in accordance with claim **1**, wherein the predetermined information element specifies at least one additional information element, and the method further comprises the steps of:

at the requesting communication unit:

(l) transmitting a request for said at least one additional information element to the responding communication unit;

at the intervening communication unit:

(m) intercepting the request for said at least one additional information element;

(n) extracting search-specific information regarding said at least one additional information element from the intercepted request;

(o) conducting a series of searches for said at least one additional information element within a predetermined set of target libraries, wherein library search chronology and specific target library designation are determined by said extracted search-specific information; and

(p) transmitting said at least one additional information element to the requesting communication unit.

9. The method in accordance with claim **8**, wherein the step (l) of transmitting a request for said at least one additional information element comprises the step of transmitting a request for at least one additional document referenced in the originally requested predetermined information element.

10. The method in accordance with claim **8**, wherein the step (n) of extracting search-specific information comprises the step of examining document class designations for said additional documents referenced.

11. The method in accordance with claim **8**, wherein the step (o) of conducting a series of searches further comprises the steps of:

(o1) first, searching document libraries on local storage media with order of search determined at least in part by document class designation; and

(o2) subsequently searching document libraries associated with remote communication units.

12. Apparatus for exchange of information comprising:

means for transmitting a request for a predetermined information element from a requesting communication unit to a responding communication unit using a first format;

means for intercepting the request for the predetermined information element at an intervening communication unit;

means for searching resources local to the intervening communication unit for the requested predetermined information element;

means for determining whether the requested predetermined information element is located in local resources;

means for transmitting the requested predetermined information element to the requesting communication unit if the requested predetermined information element is located in local resources;

means for translating the request for the predetermined information element into a second format provided that the requested predetermined information element is not located in local resources;

means for transmitting the translated request to the responding communication unit;

means for receiving the translated request at the responding communication unit;

means for transmitting a representation of the requested predetermined information element to the requesting communication unit;

means for intercepting the transmitted representation of the requested predetermined information element at the intervening communication unit;

means for expanding the representation to provide the requested predetermined information element; and

means for transmitting the requested predetermined information element to the requesting communication unit.

13. The apparatus of claim **12**, wherein the means for transmitting a request comprises a requesting computer system interconnected with the responding communication unit.

14. The apparatus of claim **13**, wherein the requesting computer system includes web browser software.

15. The apparatus of claim **12**, wherein the means for intercepting the request comprises an intervening communication unit interconnected with the requesting communication unit.

16. The apparatus of claim **15**, wherein the intervening communication unit comprises a communication unit selected from the group of communication units consisting of:

a file server;

a proxy server; and

a common gateway interface.

17. A method for implementing an abridged communication protocol for exchange of information among communication units, wherein a designated server communication unit includes, accessible in local storage, a document directory having a symbolic reference to available documents, selected parameter values, and selected operations, the method comprising the steps of:

at a requesting communication unit:

(a) transmitting a request to the designated server communication unit for a predetermined document defining selected protocol standards; and upon receiving said predetermined document,

(b) formatting a request in accordance with the protocol standards; and

(c) transmitting the request to the designated server communication unit;

at the designated server communication unit:

(d) receiving the transmitted request;

(e) extracting request-specific information from the received request; and, if the request is for a document,

(f) using a received document identifier as an index into the symbolic reference to available documents, accessing and transmitting the requested document to the requesting communication unit; and if the request is for a parameter value,

(g) using a received parameter identifier as an index into the symbolic reference to selected parameter values, accessing and transmitting the requested parameter to the requesting communication unit; and, if the request is for an operation,

(h) using a received operation identifier as an index into the symbolic reference to selected operations, performing the selected operation.

18. The method in accordance with claim **17**, wherein the abridged communication protocol comprises an enhanced version of HTTP.

19. The method in accordance with claim **17**, wherein the step (a) of transmitting a request to the designated server communication unit comprises transmitting a request for a capabilities document that includes indicia representing software version and revision level, ordering of data elements, and size of data elements.

20. The method in accordance with claim **17**, wherein the symbolic reference to available documents comprises a document directory having a symbol table including entries corresponding to documents, parameter values and operations stored on or recognized by the server communication unit.

21. The method in accordance with claim **20**, wherein documents stored on or recognized by the server communication unit are of a type selected from the set of document types consisting of:

static documents;

dynamic documents;

invocation documents; and

variable manipulation documents.

22. The method in accordance with claim **21**, wherein static documents comprise documents that change only when software version or revision level changes.

23. The method in accordance with claim **21**, wherein dynamic documents comprise documents that change from time to time, and each dynamic document includes a stored modification date and a stored revision date.

24. The method in accordance with claim **21**, wherein an invocation document invokes an operation.

25. The method in accordance with claim **21**, wherein a variable manipulation document manipulates a parameter value.

26. The method in accordance with claim **19**, wherein the capabilities document is the first ordered document stored on the server and is denominated document 0.

27. The method in accordance with claim **20**, wherein the document directory is the second ordered document stored on the server and is denominated document 1.

28. The method in accordance with claim **20**, wherein the symbolic reference to selected parameter values is the third ordered document stored on the server and is denominated document 2.

* * * * *

X. RELATED PROCEEDINGS

None.